ulm university universität ulm

Universität Ulm | 89069 Ulm | Germany

Fakultät für Ingenieurwissenschaften, Informatik und Psychologie
Institut für Datenbanken und Informationssysteme

Konzept und Realisierung einer Plattform zur Extraktion von Fragebogendaten für Analysesysteme

Bachelorarbeit an der Universität Ulm

Vorgelegt von:
Rita Unseld
rita.unseld@uni-ulm.de

Gutachter:
Prof. Dr. Manfred Reichert

Betreuer:
Dr. Johannes Schobel

2018

Fassung 9. Dezember 2018

Kurzfassung

Studien und Umfragen dienen dazu, Analysten möglichst umfangreiche und verlässliche Datenbasen für ihre Statistiken zur Verfügung zu stellen. In zunehmendem Maße werden diese Daten computergestützt erhoben und verifiziert. Der Einsatz mobiler Endgeräte kann dabei sowohl qualitativen als auch quantitativen Nutzen bringen.

Intelligente Fragebogen-Anwendungen, die den Anwender beim Ausfüllen des Fragebogens unterstützen, müssen nicht nur die technischen Anforderungen der Hard- und Software von mobilen Endgeräten berücksichtigen, sondern auch den inhaltlichen Vorgaben von Domänen-Experten, wie Medizinern oder Biometrikern, entsprechen. Um dem Domänen-Experten die eigenständige Erstellung, Verteilung und Anpassung solcher Fragebogen-Anwendungen zu ermöglichen, wurde an der Universität Ulm das QuestionSys Framework entwickelt. Mit diesem sind Domänen-Experten in der Lage, ohne die Mitwirkung von IT-Experten Fragebogen-Anwendungen für mobile Endgeräte zu generieren und die erhobenen Daten für Analysezwecke zu exportieren. Das QuestionSys Framework stellt die Daten in Form von Javascript Object Notation-Dokumenten zur Verfügung. Diese enthalten neben den reinen Ergebnisdaten eines Fragebogens auch zahlreiche Informationen über dessen Ausfüllprozess.

Das Ziel der vorliegenden Arbeit ist die Konzeption und Realisierung einer Plattform zur Extraktion und Aufbereitung der Fragebogendaten für die statistische Analyse. Zu diesem Zweck wurde untersucht, welche dieser Daten extrahiert werden sollen und wie sie aufbereitet werden müssen. Aus den Erkenntnissen des experimentellen Prototypings und aus Gesprächen mit Statistikern der Universität Ulm wurden die technischen und inhaltlichen Anforderungen an die Extraktionsplattform abgeleitet. Die Umsetzung erfolgte in Form einer Webanwendung.

Die Extraktion und Bereitstellung der Fragebogendaten wurde exemplarisch für die Statistik-Software R realisiert. Um die Extraktionsplattform auch für andere Zielsysteme nutzen zu können, wurde bei der Planung besonderer Wert auf die Erweiterbarkeit des Softwaresystems gelegt.

Danksagung

Ich danke Dr. Johannes Schobel für seine hilfreichen Anregungen und die konstruktive Kritik bei der Konzeption und Umsetzung dieser Arbeit sowie für die Motivation, mit der er mich für dieses Thema begeistert hat.

Mein weiterer Dank gilt Dr. Hartmut Lanzinger und meiner Tochter Theresa Unseld, die mich als potenzielle Endanwender im Bereich der statistischen Analyse beraten haben.

Inhaltsverzeichnis

1

Einleitung

Statistiken begleiten unser tägliches Leben. Oft werden die dazu benötigten Daten in Form von papierbasierten Fragebögen bei Ärzten, Evaluationen, Marktforschungen u.v.a erhoben. Zur computergestützten Analyse müssen die erhobenen Daten in elektronischer Form vorliegen. Dazu werden sie erfasst, validiert und in das benötigte Format überführt.

Das manuelle Ausfüllen eines Papierfragebogens ist fehleranfällig. Die Angaben können unleserlich, offensichtlich falsch oder inkonsistent sein. Des weiteren ist die manuelle oder computergestützte Erfassung der Ergebnisse von Papierfragebögen zeit- und personalaufwendig. Sie müssen gesammelt, zentral eingescannt und gegebenenfalls ergänzt oder korrigiert werden. Erst dann können die Daten in ein Analysesystem übernommen und ausgewertet werden [1].

Eine Möglichkeit, diesem Problem aus Sicht der Informatik zu begegnen, könnte sein, das Ausfüllen des Fragebogens durch entsprechende Soft- und Hardware zu unterstützen [2]. Zu diesem Zweck wurde am Institut für Datenbanken und Informationssysteme der Universität Ulm anhand von 8 ausgewählten Projekten untersucht, ob und wie das Ausfüllen von Fragebögen auf mobilen Endgeräten die o.g. Probleme verringern könnte.

Alle Projekte ergaben, dass die Datenqualität der Antworten signifikant höher war als mit papierbasierten Fragebögen, was darauf zurückzuführen ist, dass diese bereits beim Ausfüllen des Fragebogens auf Korrektheit und Konsistenz geprüft wurden. Auch die Quantität der erfassten Daten war beachtlich. So wurden in einem weltweiten Projekt zur Erfassung von Tinnitus Symptomen innerhalb eines Jahres mehr als 20.000 Befragungen mit über 200.000 resultierenden Ergebnisdaten durchgeführt [3].

Die an diesen Projekten beteiligten Domänen-Experten äußerten zudem den Wunsch, die computergestützten Fragebögen mehrsprachig und eigenständig, d.h. ohne die Mitwirkung eines IT-Experten erstellen zu können.

Die Erkenntnisse aus diesen Projekten und die Anregungen der Domänen-Experten führten zur Entwicklung des QuestionSys Frameworks. Dieses unterstützt den Endbenutzer bei der Generierung von Fragebogen-Anwendungen für mobile Endgeräte und liefert qualitativ hochwertigere Daten, indem es diese bereits beim Erfassen auf Konsistenz und logische Korrektheit prüft.

1.1 Problemstellung und Zielsetzung

Das QuestionSys Framework stellt über eine Web-Schnittstelle die zu einem Fragebogen gespeicherten Antworten in Form der JavaScript Object Notation (JSON) zur Verfügung. Im QuestionSys Framework liegen die Daten der Fragebögen und ihrer Antworten in geschachtelten und verschlüsselten Strukturen vor. Um sie mit Hilfe eines Analysesystems auswerten zu können, müssen sie extrahiert, entschlüsselt und in deren Format transformiert werden.

Die im Rahmen der vorliegenden Abschlussarbeit entwickelte Extraktionsplattform soll dem Analysten helfen, die für ihn relevanten Daten aus den vom QuestionSys Framework bereitgestellten JSON-Daten zu extrahieren, zu interpretieren und anschließend in das für sein Analysesystem benötigte Format zu transformieren. Dies ist die zentrale Funktionalität der Extraktionsplattform. Exemplarisch wurde sie für die Statistiksoftware R realisiert, wobei eine möglichst einfache Erweiterung für weitere Analysesysteme wie Excel, SAS oder SPSS bereits in der Konzeption berücksichtigt wurde.

1.2 Struktur der Arbeit

Kapitel 2 stellt die Funktionsweise des QuestionSys Framework vor, erklärt Aufbau und Inhalt der JSON-Daten und beschreibt die Grundlagen ihrer Verarbeitung in der

Statistiksoftware R. Kapitel 3 analysiert die inhaltlichen und technischen Problemquellen der Extraktionsplattform und definiert die sich daraus ergebenden funktionalen und nichtfunktionalen Anforderungen.

Das 4. Kapitel enthält das Konzept für die System-Architektur und die Funktionen der zu erstellenden Software. Kapitel 5 beschreibt, wie dieses Konzept realisiert wurde. In Kapitel 6 werden alternative Anwendungen für die Generierung von Online-Fragebögen und deren Form der Berichterstellung diskutiert. Im 7. Kapitel werden die gewonnenen Erkenntnisse zusammengefasst und ein Ausblick auf mögliche Weiterentwicklungen gegeben.

Um dem Leser das Verständnis für die sehr komplexe Struktur der zu extrahierenden JSON-Daten zu vereinfachen, führt ein Beispiel-Fragebogen durch die gesamte Abschlussarbeit. Dieser wird in Kapitel 2.2.3 vorgestellt und in den nachfolgenden Kapiteln zur Veranschaulichung der jeweiligen Thematik wieder aufgegriffen.

2

Grundlagen

Dieses Kapitel führt in die Funktionsweise des QuestionSys Frameworks ein und erläutert, in welcher Form es die Daten eines Fragebogens für die Extraktionsplattform zur Verfügung stellt. Dazu gibt es eine Einführung in die JavaScript Object Notation (JSON) und zeigt anhand eines Beispiels, wie die Fragebogen-Daten im JSON-Format gespeichert werden. Es folgen einige Informationen zur Statistik-Software R und dem R-Paket jsonlite, mit dessen Hilfe JSON-Daten in eine R-Datenstruktur überführt werden kann.

2.1 QuestionSys Framework

Das QuestionSys Framework (siehe Abbildung 2.1) ist eine Anwendung zur digitalen Erhebung und Bereitstellung von Daten zum Zweck der statistischen Analyse. Der Domänen-Experte, z.B. ein Mediziner oder Psychologe, generiert mittels eines graphischen Konfigurators das mit Prozessen hinterlegte Modell eines Online-Fragebogens ⓐ. Aus diesem erstellt das Framework ohne die Mitwirkung eines IT-Experten digitale, prozessgesteuerte Fragebogen-Anwendungen, die auf mobilen Endgeräten ausgeführt werden können ③. Die bei der Ausführung dieser Anwendungen gesammelten Daten werden in Ausführungsdateien (`Execution Log Files`) ④ zentral gespeichert.

Für die Auswertung dieser Daten stellt eine Webschnittstelle des Questionsys Frameworks dem berechtigten Benutzer, z.B. einem Analysten, anonymisierte Ausführungsdaten (`Anonymized Execution Log Files`) ⑤ im JSON-Format zur Verfügung. Sie stellen die zentrale Datenbasis für die in dieser Abschlussarbeit entwickelte Extraktionsplattform dar und werden in Kapitel 2.2 näher erläutert. Mit Hilfe der Extraktionsplattform

soll der Analyst die für ihn relevanten Daten aus den anonymisierten Ausführungsdaten extrahieren und in das für seine Statistik-Software benötigte Datenformat umwandeln können ⓑ [3].

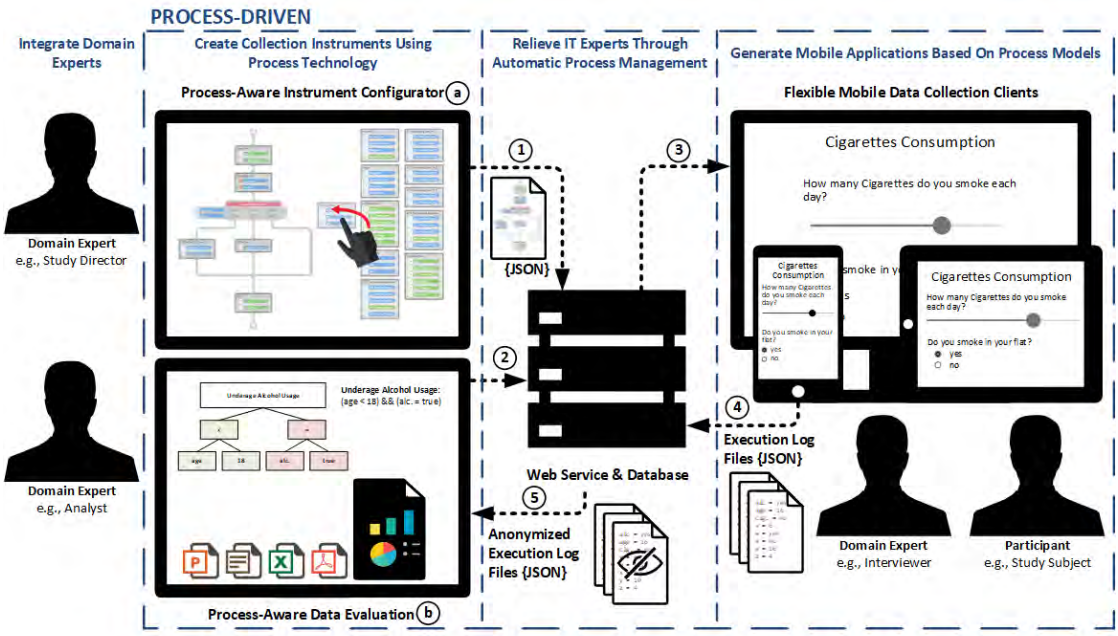


Abbildung 2.1: Architektur des QuestionSys Frameworks [3]

2.2 Anonymisierte Ausführungsdaten

Das QuestionSys Framework setzt JSON-Dokumente zur Kommunikation zwischen den Komponenten und zur Speicherung der Daten ein. Auf Anfrage erstellt eine Webschnittstelle des QuestionSys Frameworks zwei JSON-Dateien, welche die strukturellen und inhaltlichen Daten des Online-Fragebogens und der zugehörigen anonymisierten Ergebnisse enthalten. Diese Daten bilden die in Abbildung 2.1 dargestellten `anonymisierten Ausführungsdaten` ⑤.

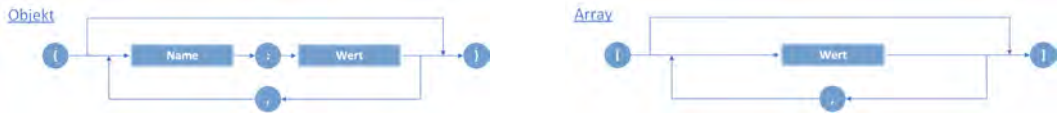
Die nachfolgenden Kapitel geben eine Einführung in den grundsätzlichen Aufbau von JSON-Daten und beschreiben die Repräsentation der anonymisierten Ausführungsdaten in diesem Format.

2.2.1 JSON und seine Datenstrukturen

JSON ist ein Text-Format, welches im Web zum Austausch und zur Speicherung strukturierter Daten verwendet wird. JSON, welches heute sprach- und plattform-unabhängig ist, hat seinen Ursprung in den JavaScript Definitionen für Objekte und Arrays. Alle modernen Programmiersprachen (z.B. Java, C#, Python u.v.a.) können JSON-Daten erstellen (serialisieren) und lesen (deserialisieren). Aktuell wird JSON durch zwei Standards spezifiziert, durch IETF [1] RFC 8259 von Douglas Crockford und durch ECMA[2] 404 [4] [5].

Ein Objekt in JSON ist eine ungeordnete Menge von Schlüssel/Wert Paaren (siehe Abbildung 2.2a). Die Elemente dieser Menge sind über ihren Schlüssel eindeutig referenzierbar. Der zugehörige Wert kann dabei wiederum Objekt, Array, Zeichenkette, Zahl oder boolescher Wert sein. Dabei ist es nicht notwendig, dass alle Werte vom selben Typ sind. Man erkennt ein Objekt an den es umschließenden geschweiften Klammern.

Ein Array stellt eine geordnete Liste von Werten dar (siehe Abbildung 2.2b). Ihre Elemente werden über ihren Index eindeutig referenziert. Auch diese Werte müssen nicht vom selben Typ sein. Kennzeichen eines Arrays sind die umschließenden eckigen Klammern.



(a) JSON Objekt (b) JSON Array

Abbildung 2.2: JSON Datenstrukturen

2.2.2 Fragebogen und Ausführungsdaten als JSON-Objekte

Der Fragebogen und seine Ausführungsdaten werden der Extraktionsplattform als JSON-Objekte zur Verfügung gestellt. Das JSON-Objekt eines Fragebogens enthält umfangreiche Informationen über dessen Inhalt und Struktur. Für die Extraktionsplattform sind neben dem Namen und den Sprachen des Fragebogens vor allem die Inhalte und Typen

[1] Internet Engineering Task Force
[2] European Computer Manufacturers Association (heute ECMA International)

seiner Fragen relevant. Ein Fragebogen enthält beliebig viele Fragen mit unterschiedlichen Fragetypen. Diese sind ausschlaggebend dafür, wie die Fragen im Fragebogen repräsentiert und in den Ausführungsdaten gespeichert werden. Tabelle 2.1 zeigt, welche Fragetypen aktuell im QuestionSys Framework verwendet werden.

Tabelle 2.1: QuestionSys Fragetypen

Fragetypen	
1.1 SingleChoice	Nur ein Element ist auswählbar
1.2 Yes No	Nur ein Element ist auswählbar
2.1 Buttongrid	Mehrere Elemente sind auswählbar
2.2 MultipleChoice	Mehrere Elemente sind auswählbar
3.1 Ranking	Elemente werden nach Vorliebe eingeordnet
3.2 Distribution	Punkte werden auf vorgegebene Elemente verteilt
3.3 Matrix	Aussagen in Reihen und Spalten werden verknüpft
4.1 SliderRange	Anfang und Ende eine Spannweite
4.2 SliderSingle	Ein Wert aus einem Bereich ist auswählbar
5.1 FreeDate	Individuelle Datums-Eingabe
5.2 FreeTextarea	Individuelle Text-Eingabe
5.3 FreeFloat	Individuelle Fließkommazahl-Eingabe

Abbildung 2.3 zeigt ein stark vereinfachtes Domänenmodell des Fragebogens und seiner Ausführungsdaten. Alle `Fragen` sind Bestandteil des Fragebogen-Objekts. Sie alle beinhalten beschreibende Informationen und Vorgaben für Gültigkeitsprüfungen. Die Fragetypen 1.1 bis 3.3 enthalten Arrays von `Elementen`. Diese besitzen einen eindeutigen Schlüssel und einen beschreibenden Text in allen Sprachen des Fragebogens.

Die zuvor beschriebenen anonymisierten Ausführungsdaten enthalten u.a. die `Ergebnis`-Objekte, welche über einen Frage-Schlüssel die zugehörige Frage im Fragebogen-Objekt referenzieren. Die zu den Fragetypen 1.1 bis 3.3 gehörenden Ergebnis-Werte werden verschlüsselt abgespeichert. D.h. ein Element-Schlüssel in den Ergebnissen zeigt auf das ausgewählte Element der Frage im Fragebogen. Die Werte der restlichen Ergebnisse der anderen Fragetypen liegen codiert oder im Klartext vor.

Das nachfolgende Kapitel veranschaulicht anhand eines Beispiels, wie das QuestionSys Framework einen Fragebogen mit einer MultipleChoice Frage und ein zugehöriges Ergebnis als JSON-Objekte speichert.

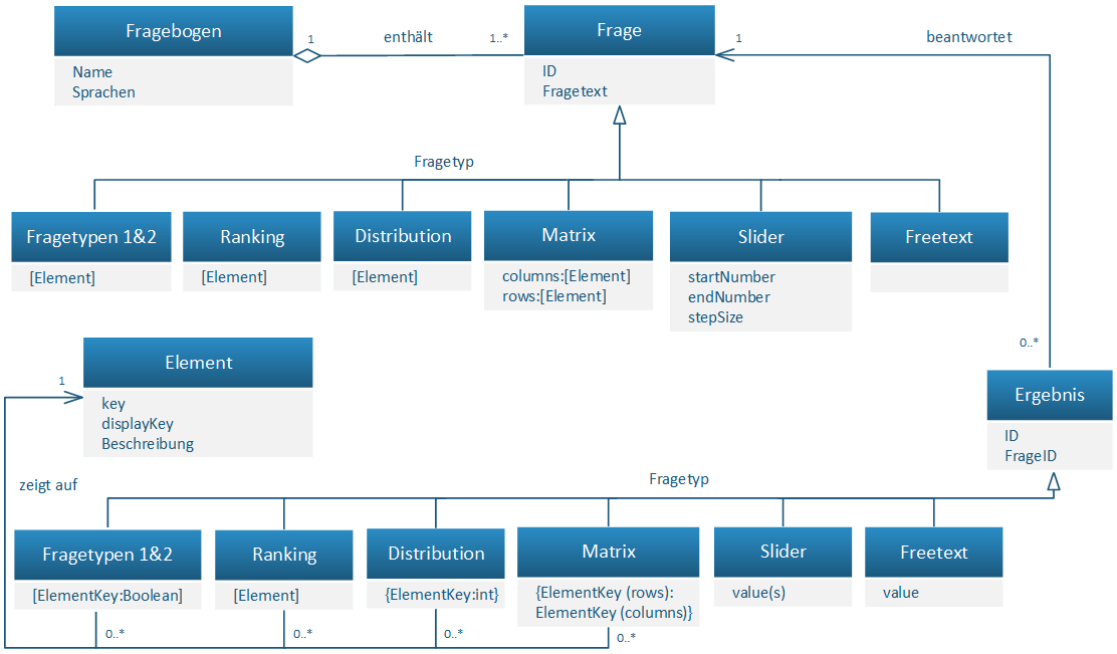


Abbildung 2.3: Domänenmodell der Fragebogen- und Ausführungsdaten

2.2.3 Beispiel

Als durchgängiges Beispiel dieser Abschlussarbeit dient ein Fragebogen, welcher das Konsumverhalten von Technik-Produkten erheben soll. Abbildung 2.4 zeigt in graphischer Form die JSON-Objekte, welche die Fragebogen-Struktur sowie die Ergebnisse eines ausgefüllten Fragebogens enthalten. Wie eingangs erläutert, können JSON-Objekte beliebig verschachtelt sein und neben einfachen Datentypen auch weitere Objekte enthalten.

Das Objekt `meta` beinhaltet u.a. den Namen des Fragebogens und die Sprachen, in denen dieser zur Verfügung steht. Die Fragen, deren Typ und mögliche Antworten speichert das `model` im Array `nodeDataArray`. Das Beispiel zeigt eine Frage vom Typ MultipleChoice, welche durch den key `-13` eindeutig identifiziert wird. Die Frage ist in zwei Sprachen verfügbar. Die möglichen Antworten (`Elemente`) sind im Array `items` abgelegt. Jedes Element enthält einen beschreibenden Text in allen Sprachen des Fragebogens und einen identifizierenden `key`.

Die anonymisierten Ergebnisdaten bestehen aus einem Array (`data`) von JSON-Objekten, welche jeweils einen ausgefüllten Fragebogen repräsentieren. Diese wiederum enthalten ein Objekt `results`, welches jeder Frage aus dem Fragebogen-Objekt die vom Benutzer gewählten Elemente zuordnet. Dabei stellt im Beispiel der key `-13` die Verbindung zwischen der Frage und ihren Antworten her. Die gegebene Antwort ergibt sich aus dem mapping des `items` im Frage-Objekt mit dem `value` des Ergebnis-Objekts.

Im konkreten Fall heißt das, der Benutzer hat auf die Multiple Choice-Frage „*Besitzen Sie eines oder mehrere der folgenden Technik-Produkte?*“ geantwortet, dass er ein Smartphone besitzt.

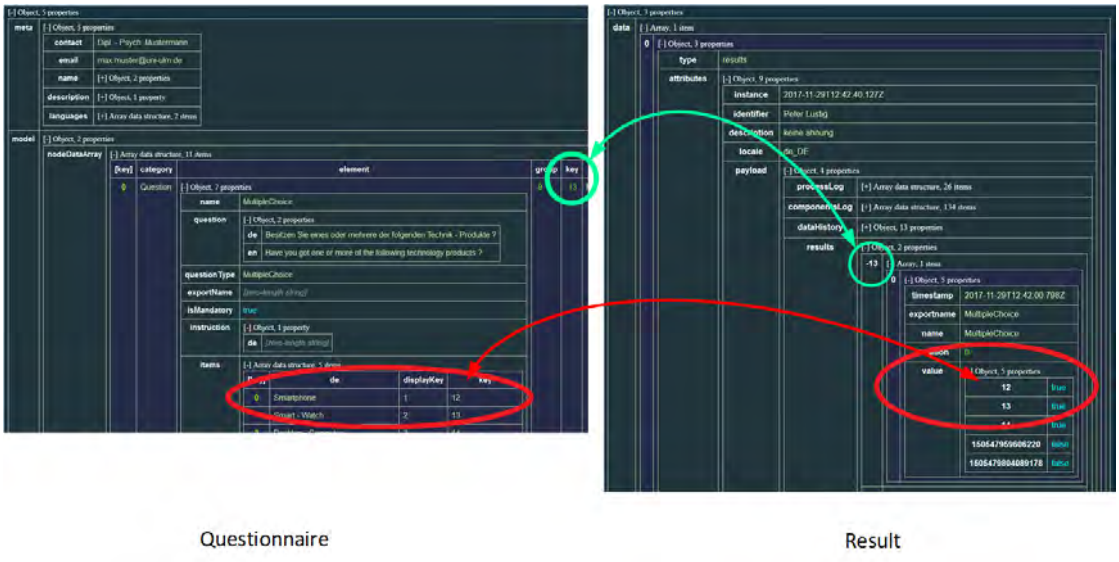


Abbildung 2.4: graphische Darstellung von Fragebogen und Ergebnis

2.3 JSON und R

Die vorliegende Abschlussarbeit beschreibt die Extraktion der JSON-Daten für R, eine Open-Source-Software für statistische Berechnungen und Graphiken. Die Wahl fiel auf R, da sich diese in den letzten Jahren zur populärsten Software für statistische Datenanalysen entwickelt hat. Sie wurde 1992 an der Universität Auckland entwickelt und zeichnet sich u.a. dadurch aus, dass eine weltweite Community an Programmierern und

Statistikern die Funktionalität ständig erweitert und diese Erweiterungen als sogenannte Pakete frei zur Verfügung stellt [6].

Für die Transformation der JSON-Daten benötigt R einen Parser. Ein Parser ist ein Algorithmus oder Programm, das einem formalsprachlichen Ausdruck einen anderen formalsprachlichen Ausdruck zuordnet [7]. Im Fall von JSON ersetzt man *Ausdruck* durch *Datenstrukturen*. Mit dem R-Paket `jsonlite` können JSON-Daten aus einer Datei oder über eine URL von einer Web-Seite in R geladen werden. Theoretisch könnte ein Analyst direkt mittels eines JSON-Parsers in seiner Statistiksoftware auf die Ausführungsdaten zugreifen. Um eine saubere Datenbasis zu erhalten, welche die Grundlage analytischer Prozesse bildet [8], müsste er dann die verschlüsselten Informationen selbst extrahieren und entschlüsseln.

Kapitel 3 analysiert, welche Problemquellen berücksichtigt werden müssen, um eine saubere Datenbasis zu erhalten und welche Anforderungen an die Extraktionsplattform sich daraus ergeben.

3

Anforderungsanalyse

Auf Basis der im vorigen Kapitel beschriebenen Grundlagen wurde für die Extraktionsplattform ein experimenteller Prototyp entwickelt, welcher in mehreren Evaluierungsschritten erweitert und an die Bedürfnisse der potentiellen Benutzer angepasst wurde.

> Prototypen sollten immer dann entwickelt werden, wenn wichtige Anforderungen fehlen oder nur vage und unvollständig formuliert werden können. [9]

Beim Prototyping existieren verschiedene Vorgehensweisen. In allen dient jedoch der Prototyp zum Bestimmen und Konkretisieren der Anforderungen an ein Softwaresystem. Das explorative Prototyping bspw. konzentriert sich vor allem auf die Erarbeitung einer evaluierten und daher belastbaren Anforderungsspezifikation der Softwarefunktionalität. Beim experimentellen Prototyping, welches in dieser Abschlussarbeit angewendet wurde, liegt der Fokus auf der Suche nach Möglichkeiten zur technischen Umsetzung eines Entwicklungsziels [10]. Durch Experimente und Evaluierungen des Prototypen wurden die Anforderungsdefinitionen für die Extraktionsplattform erarbeitet und die technische Realisierung erforscht. Sowohl beim explorativen als auch beim experimentellen Prototyping wird der entwickelte Prototyp am Ende verworfen und die Software nach den gewonnenen Erkenntnissen neu entwickelt.

So konnten sukzessive die nachfolgenden funktionalen und nichtfunktionalen Anforderungen herausgearbeitet werden. Diese sind zuerst allgemein formuliert. Sie werden anschließend durch konkrete Anforderungen ergänzt, die sich während des Prototypings ergeben haben.

3.1 Funktionale Anforderungen

Dieses Kapitel beschreibt, welche Funktionalitäten die Extraktionsplattform dem Benutzer zur Verfügung stellen soll.

FA1 Die Extraktionsplattform soll dem Benutzer die ersten 20 Ergebnisse des Fragebogens in entschlüsselter Form anzeigen. Die dazu benötigten JSON-Objekte erhält die Extraktionsplattform von einer Web-Schnittstelle des QuestionSys Frameworks.

FA2 Die Extraktionsplattform soll dem Benutzer auf Knopfdruck die Struktur und die Dekodierungstabelle des Fragebogens anzeigen.

FA3 Die Extraktionsplattform soll dem Benutzer eine druckbare Version der Dekodierungstabelle anzeigen, wenn er die Webbrowser Druckfunktion verwendet.

FA4 Der Benutzer soll die Möglichkeit haben, diejenigen Variablen aus den Ausführungsdaten zu selektieren, die er in seine Analyse-Anwendung extrahieren möchte.

FA5 Die Extraktionsplattform soll dem Benutzer anzeigen, in welchen Sprachen der Fragebogen generiert wurde, und ihm ermöglichen, eine der Sprachen für die Darstellung und Extraktion auszuwählen.

FA6 Der Benutzer soll über die Extraktionsplattform auswählen können, für welche Analyse-Anwendung die Ausführungsdaten extrahiert werden soll.

FA7 Die Extraktionsplattform soll auf Knopfdruck für die ausgewählten Variablen und die ausgewählte Statistiksoftware ein Extraktionsskript generieren. Dieses soll die über eine Webschnittstelle bereitgestellten Ausführungsdaten in das benötigte Format extrahieren und dekodieren.

3.2 Nichtfunktionale Anforderungen

Die nichtfunktionalen Anforderungen spezifizieren die Qualitätsattribute (QA) des zu entwickelnden Software-Systems. Nach Sommerville [11] und dem IEEE-Standard 29148 [12] sind die unten aufgeführten Qualitätsanforderungen QA1 bis QA6 ungenügend, da

sie nicht verifizierbar sind. Darauf wurde bewusst verzichtet, da eine Erfüllung der nichtfunktionalen Anforderungen im Rahmen dieser Abschlussarbeit nicht effektiv überprüfbar ist.

Des weiteren sind hier nur diejenigen Qualitätsattribute aufgeführt, die für die Wartbarkeit[1] der Extraktionsplattform besondere Beachtung finden müssen. Sowohl das QuestionSys Framework als auch die Extraktionsplattform befinden sich aktuell noch in einer Entwicklungsphase. Daher kommt der Wartbarkeit der Software eine gesteigerte Bedeutung zu.

Die üblichen Qualitätsanforderungen an gute Software werden als obligatorisch vorausgesetzt.

QA1 Die Systemarchitektur der Extraktionsplattform soll die Forderung nach guter Wartbarkeit berücksichtigen.

QA2 Die Extraktionsplattform soll den Coding Conventions derjenigen Programmiersprache entsprechen, in der sie umgesetzt wird, um eine gute Wartbarkeit zu erreichen.

QA3 Die Extraktionsplattform soll so flexibel umgesetzt sein, dass sie ohne Änderungen des Gesamtkonzepts um ein zusätzliches Analysesystem oder einen zusätzlichen Fragetyp erweitert werden kann.

QA4 Die Extraktionsplattform soll im Fall eines auftretenden Fehlers eine entsprechende Fehlermeldung anzeigen und dem Benutzer die Rückkehr auf die Startseite ermöglichen.

QA5 Die graphische Benutzeroberfläche der Extraktionsplattform soll selbsterklärend sein.

QA6 Das generierte Extraktionsskript soll die Forderung nach guter Performance berücksichtigen.

[1] Die Wartung eines Softwareprodukts beinhaltet u.a. Veränderungen zur Funktionserhaltung und zur Funktionserweiterung.

3.3 Anforderungen an den JSON-Parser

Wie in Kapitel 2 beschrieben, wurde entschieden, die Ausführungsdaten des QuestionSys Frameworks zunächst für die Statistiksoftware R zu extrahieren. Es gibt drei bekannte R-Pakete, die JSON-Daten in ein R Format überführen können. Dies sind: `rjson`, `RJSONIO` und `jsonlite`. Von diesen drei Paketen ist allerdings nur `jsonlite` in der Lage, direkt mit einer Web-Schnittstelle zu kommunizieren. Daher kann für die funktionale Anforderung FA7 nur dieses Paket eingesetzt werden.

Außerdem ist `jsonlite` für die Arbeit mit großen Datenmengen aus dem Web optimiert [13]. Wenn die Größe der Ausführungsdaten es nicht zulässt, diese in einem einzigen Request an den Client zu übertragen, werden sie von der Webschnittstelle fragmentiert, ähnlich wie die Seiten eines Buchs. Um diese Daten in R analysieren zu können, müssen die Fragmente wieder zu einem Datensatz zusammengefügt werden. `jsonlite` bietet dazu die Funktion `rbind_pages`.

3.4 Konkrete Anforderungen an die R Daten

Mit Hilfe des R-Pakets `jsonlite` können JSON-Daten direkt in eine R-Datenstruktur geladen werden. JSON, sowie die meisten Programmiersprachen und Datenbankanwendungen, ist für zeilenbasierte, R jedoch für spaltenbasierte Operationen optimiert. Diese Diskrepanz führt zu verschiedenen Komplikationen bei der Umwandlung und Auswertung der JSON-Daten in R. Nachfolgend wird beschrieben, wie sich diese Komplikationen äußern und welche konkreten Anforderungen sich daraus für die Extraktionsplattform ergeben haben.

Bei den Ausführungsdaten des QuestionSys Frameworks handelt es sich um hierarchisch stark verschachtelte Objekte, die bei der Umwandlung in R ebenfalls eine stark verschachtelte Datenstruktur aus Listen, Data Frames[2], Vektoren und einfachen Datentypen erzeugen. Abbildung B.2 visualisiert die Transformation der Ausführungsdaten

[2] „A data frame is a list of variables of the same number of rows with unique row names, given class *data.frame*“ [14]

des Beispiel-Fragebogens aus Kapitel 2. Daran lässt sich erkennen, wie `jsonlite` versucht, aus dem zeilenbasierten JSON-Array ein spaltenbasiertes R Data Frame zu generieren. Es fasst dazu die Zeilen mit den Ergebnissen der MultipleChoice Frage mit dem Schlüssel `-13` zu einer Spalte in R zusammen. Dabei entsteht allerdings wiederum eine Liste einzelner Data Frames. R muss diese Liste für eine Auswertung der Ergebnisdaten zeilenweise abarbeiten, um an die Werte innerhalb jedes Data Frames zu gelangen. Abbildung 3.1 zeigt den Inhalt des Data Frames für das erste Ergebnis der MultipleChoice Frage. Eine solche Liste verschachtelter Daten ist ungeeignet und ineffizient für eine Analyse in R [13].

```
> result_data$data$attributes$payload$results$`-13`
[[1]]
      exportname iteration          name             timestamp value.12 value.13 value.14 value.150547959606220 value.1505479804089178
1 MultipleChoice         0 MultipleChoice 2017-11-29T12:39:11.825Z    FALSE    FALSE     TRUE                 TRUE                  FALSE
```

Abbildung 3.1: Ergebnis der MultipleChoice Frage als Data Frame

Eine saubere Tabellenstruktur, die eng mit den Prinzipien relationaler Datenbanken und Codd's Relationen Algebra verbunden ist, erleichtert und beschleunigt dagegen die Analyse in R. Hadley Wickham, Chief Scientist bei RStudio beschreibt in seinem Papier `Tidy Data` wie ein sauberer R-Datensatz aussehen sollte und welche Transformationen dazu nötig sind[15][16].

Demnach sind die folgende Kriterien Voraussetzung für einen sauberen R-Datensatz:

1. Jede Spalte entspricht genau einer Variablen. Diese enthält nur einen Wert. Alle Werte dieser Spalte sind vom gleichen Typ. Spaltenbezeichner sind Namen und keine Werte.

2. Jede Beobachtung bildet genau eine Reihe.

3. Mehrere Beobachtungen, die inhaltlich eine Einheit bilden, sind in einem Datensatz zusammengefasst.

Für die Anforderungen an das von der Extraktionsplattform zu generierende R-Skript folgt daraus:

1. Jede Frage des Fragebogens bildet eine Variable und erhält einen eindeutigen Namen. Fragen mit mehreren Antwortmöglichkeiten werden aufgespalten oder enthalten einen Wert in Form eines Vektors. Letzteres widerspricht zwar Wickhams Forderung, dass jede Variable nur einen Wert besitzen soll, ist aber ein Kompromiss, der mit potenziellen Anwendern erarbeitet wurde.
2. Jeder ausgefüllte Fragebogen, also jedes Element des `data`-Arrays, bildet eine Beobachtung.
3. Alle Beobachtungen werden zu einem gemeinsamen Datensatz zusammengefasst. Dies muss durch das R-Skript explizit geschehen, da beim Parsen der JSON-Objekte jedes Objekt ein einzelnes R Data Frame erzeugt.

4

Entwurf

Da zu Beginn dieser Abschlussarbeit nur eine ungenaue Vorstellung über die Funktionsweise der Extraktionsplattform vorhanden war, wurden die zuvor beschriebenen Anforderungen durch experimentelles Prototyping sukzessive erarbeitet. Es war daher auch nicht möglich, bereits zu Beginn einen kompletten und detaillierten Entwurf der zu implementierenden Software zu erstellen, wie es in linearen Vorgehensmodellen gefordert wird. Stattdessen wurde ein nichtlineares Vorgehen gewählt, in welchem sich Analyse, Implementierung und Evaluation zyklisch wiederholten. Hierfür genügte es anfangs, zu verstehen, wie das Produkt aussehen und was es grob leisten soll [17].

Aus den Erkenntnissen des Prototypings wurde in einem ersten konzeptionellen Schritt die Softwarearchitektur der Extraktionsplattform festgelegt. Diese definiert die Struktur und die tragenden Elemente des Softwaresystems. So gibt sie der inkrementellen Entwicklung einen Konstruktionsrahmen vor, in welchem die funktionalen und nichtfunktionalen Anforderungen umgesetzt werden.

Nachfolgend wird beschrieben, welche Überlegungen zur Auswahl der in dieser Abschlussarbeit verwendeten Softwarearchitektur führten. Ferner wird erklärt, wie diese zur Umsetzung der funktionalen und nichtfunktionalen Anforderungen beiträgt.

4.1 Softwarearchitektur

> Das Ziel der Softwarearchitektur besteht in der Minimierung der menschlichen Ressourcen, die für das Errichten und die Instandhaltung des benötigten Systems erforderlich sind. [18]

Um dieses Ziel zu erreichen, muss die Komplexität des Softwaresystems reduziert und seine Änderbarkeit erhöht werden. Komplexe Software lässt sich leichter beherrschen, indem man sie sauber modularisiert. Kopplung und Kohäsion sind Metriken, die in diesem Zusammenhang erläutert und betrachtet werden sollen [19].

- Die **Kopplung** bestimmt den Grad für die Anzahl der Verbindungen zwischen Modulen und die daraus resultierenden Abhängigkeiten. Eine hohe Kopplung erhöht die Anzahl der benötigten Schnittstellen. Daher gilt: Je geringer die Kopplung zwischen Modulen ist, desto leichter können lokale Änderungen durchgeführt werden, ohne dabei andere Module anpassen zu müssen. Vor allem zyklische Abhängigkeiten sollten vermieden werden, da sie eine getrennte Wiederverwendung der Module verhindern.
- Die **Kohäsion** bestimmt den Grad für den inneren Zusammenhang der Funktionen eines Moduls. Ein hoch kohärentes Modul enthält alle logisch zusammengehörigen Operationen und Daten, die für die Lösung einer Aufgabe benötigt werden. Dies erhöht ebenfalls die Änderbarkeit, da alle Eigenschaften, die zum Verstehen und Ändern benötigt werden, zentral an einer Stelle im Softwaresystem gespeichert sind [19].

Die Kohäsion beschreibt also die Abhängigkeiten innerhalb eines Moduls und die Kopplung die Abhängigkeiten zwischen Modulen. Diese bedingen sich gegenseitig. Ein Softwaresystem, das nur ein Modul besitzt, ist zwar minimal gekoppelt aber auch minimal kohärent. Diese beiden Metriken müssen also je nach Art und Größe des Softwaresystems gegeneinander aufgewogen werden.

Die Konzeption guter Softwarearchitekturen ist vor allem zeitaufwändig und nachträgliche Anpassungen können unter Umständen extreme Änderungskosten verursachen. Daher greifen Softwareentwickler, wenn möglich, auf etablierte Entwurfsmuster zurück. Diese beschreiben Lösungen, die bereits erfolgreich in unterschiedlichen Systemen und Umgebungen realisiert, getestet und dokumentiert wurden. [20].

Um hohe Kohäsion und lose Kopplung der Software-Module zu erreichen, wurde für die Softwarearchitektur der Extraktionsplattform das Model-View-Controller-Muster (MVC-Muster) gewählt. Es gibt für das MVC-Muster keine universelle Definition und man findet

in der Literatur diverse Umsetzungsansätze. Wie jedes Entwurfsmuster ist auch das MVC-Muster eine abstrakte Schablone, welche eine erprobte Lösung für ein Problem in einem bestimmten Kontext vorschlägt.

Wichtig bei diesem Entwurfsmuster ist vor allem die saubere Trennung von Daten, Präsentation und Logik, wodurch diese Komponenten unabhängig voneinander geändert, erweitert und wiederverwendet werden können [11]. Im MVC-Muster übernimmt das Model die Kapselung der Daten, die View deren Präsentation und der Controller die Verarbeitung der Benutzeraktionen. Diese Softwarearchitektur eignet sich besonders gut, um die unterschiedlichen Sichten der Extraktionsplattform auf die Daten zu realisieren, z.B. durch HTML-Seiten oder Extraktionsskripte.

Abbildung 4.1 zeigt, wie eine Webanwendung das MVC-Muster umsetzen kann. In dieser Form der Umsetzung verarbeitet der Controller eine Anfrage des Webbrowsers, indem er eine entsprechende Funktion des Models und die passende View aufruft. Für die Kommunikation zwischen View und Model ist er nicht zuständig. Die View fragt direkt beim Model an, und dieses übergibt seine aktuellen Daten [21]. Anders als in klassischen Desktop-Anwendungen sendet das Model nur auf Anfrage aktualisierte Daten an die View. Eine detaillierte Beschreibung der Umsetzung des MVC-Muster in dieser Abschlussarbeit folgt in Kapitel 5.2.1.

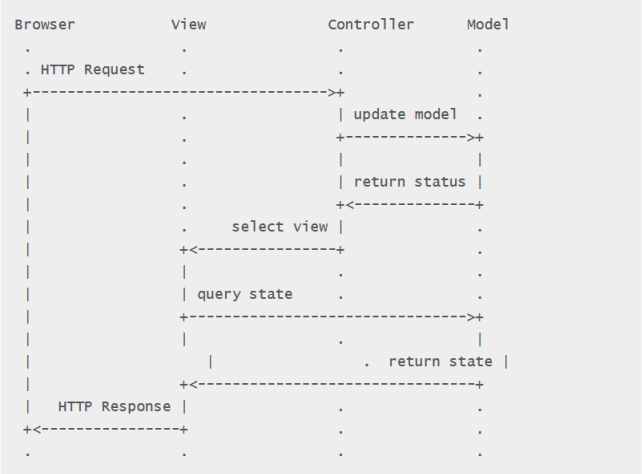


Abbildung 4.1: A sequence diagram of a single request/response pair. [21]

5

Implementierung

Dieses Kapitel beschreibt, die Umsetzung der Anforderungen aus Kapitel 3. Die Beschreibung beinhaltet die realisierten Funktionen der Extraktionsplattform, die implementierten Softwarekomponenten mit ausgewählten Codebeispielen und die Umsetzung der nichtfunktionalen Anforderungen.

Die Entwicklung der Extraktionsplattform fand außerhalb des QuestionSys Frameworks statt. Die in Kapitel 2.2 beschriebene Webschnittstelle zur Bereitstellung der Daten muss nachträglich eingebunden werden. Stattdessen wurde für die Realisierung und die Tests mit den JSON-Dateien des Beispiel-Fragebogens und der dazugehörenden Ergebnisse gearbeitet.

5.1 Funktionen der Extraktionsplattform

Abbildung 5.1 veranschaulicht die Funktionalität der Extraktionsplattform und ihre Schnittstellen zu angrenzenden Systemen. Ihre drei wesentlichen Funktionen sind die Interaktion mit dem Benutzer (User Interface), die Extraktion und Dekodierung der Ergebnisse des Fragebogens sowie die Erstellung eines Skripts zur Bereitstellung und Transformation der JSON-Daten in die Datenstruktur der ausgewählten Statistiksoftware.

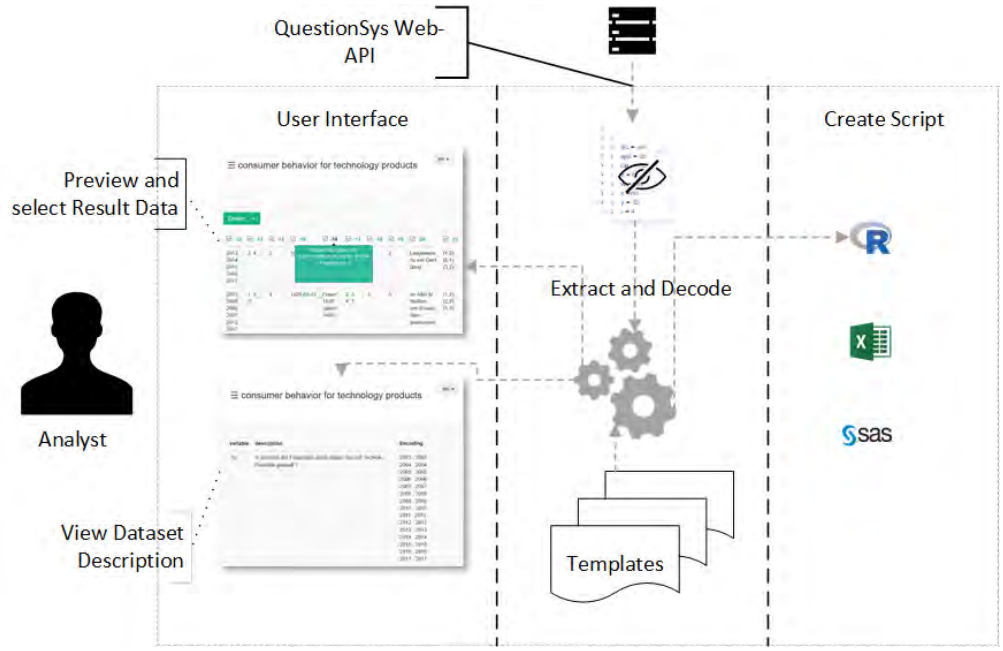


Abbildung 5.1: Funktionen der Extraktionsplattform

5.1.1 Benutzerschnittstelle

Die Benutzerschnittstelle (User Interface) der Extraktionsplattform besteht aus zwei Seiten, die in allen Sprachen des Fragebogens angezeigt werden können.

Die **Startseite** (siehe Abbildung 5.2) zeigt die ersten 20 Ergebnisse des Fragebogens, welche dem Benutzer von der Web-Schnittstelle des QuestionSys Frameworks bereitgestellt werden. So erhält er einen ersten Einblick in den Ergebnisdatensatz[1] und kann auswählen, welche Variablen[2] in seine Statistiksoftware übernommen werden sollen.

Wie in Kapitel 2 beschrieben, sind diejenigen Ergebnisdaten, für die vordefinierte Antworten hinterlegt sind, mehrfach verschlüsselt. Ein Schlüssel im Ergebnis verweist auf einen Schlüssel im Fragebogen, welcher letztlich auf die Antwortmöglichkeit zeigt. So realisiert das QuestionSys Framework u.a. sein variables Sprachen-Feature. Die Extraktionsplattform zeigt dem Benutzer für diese Fragetypen ein nur zur Hälfte dekodiertes

[1] Hinw: Statistiker verwenden den Ausdruck Datensatz für die komplette Datenmenge
[2] „Eine statistische Variable ist ein Merkmal. Der Wert einer Variablen ist die Merkmalsausprägung“ [22]

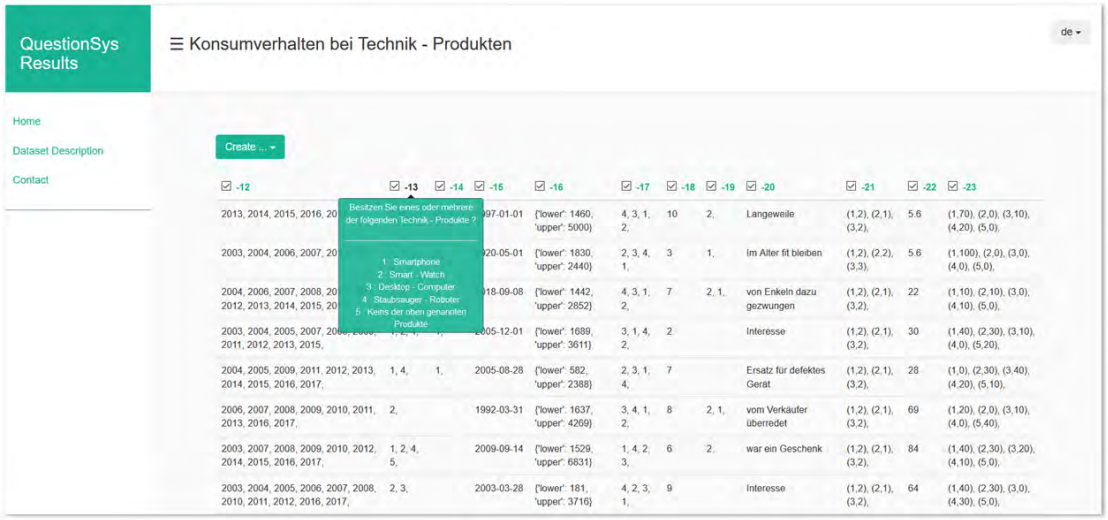


Abbildung 5.2: Anzeige der Ergebnisdaten

Ergebnis an. D.h. die Tabelle mit den Ergebnissen enthält den Schlüssel der ausgewählten Antwortmöglichkeit, nicht aber ihre textuelle Beschreibung. Statistiker arbeiten häufig mit derart kodierten Variablen und einer Beschreibung des Datensatzes in Form einer Dekodierungstabelle. Diese numerischen Variablen sind u.a. leichter zu gruppieren, zu kategorisieren und zu sortieren.

Um die auf der Startseite angezeigten Ergebnisdaten interpretieren zu können, kann sich der Benutzer ein `Tooltip` anzeigen lassen, indem er den Mauszeiger eine kurze Zeit über der entsprechenden Spaltenüberschrift verweilen lässt. Im Fall der MultipleChoice Frage aus dem Beispiel-Fragebogen lässt sich so erkennen, dass die Zahl `1` bedeutet, dass der Befragte ein *Smartphone* als Antwort ausgewählt hat. Diese HTML-Seite erfüllt die funktionalen Anforderungen FA1 und FA5.

Für die Erstellung eines Extraktionsskripts selektiert der Benutzer die Variablen, die er in seine Statistiksoftware übernehmen möchte. Dann wählt er aus, welche Statistiksoftware er verwendet und startet die Generierung des Skripts. Dieses kann daraufhin gespeichert oder direkt in der Zielsoftware geöffnet werden. Diese Funktion erfüllt die Anforderungen FA4 und FA6.

Die HTML-Seite **Datensatz-Beschreibung** (siehe Abbildung 5.3) zeigt dem Benutzer eine Beschreibung des Datensatzes in Form einer Dekodierungstabelle. Sie enthält den Variablennamen, die Beschreibung und die verschlüsselten Antwortmöglichkeiten für jede Frage des Fragebogens. Die Dekodierungstabelle wird in einer optimierten Druckversion zur Verfügung gestellt, die der Benutzer über die entsprechende Funktion seines Browsers ausdrucken kann. Auf diese Weise wurden die funktionalen Anforderungen FA2 und FA3 umgesetzt.

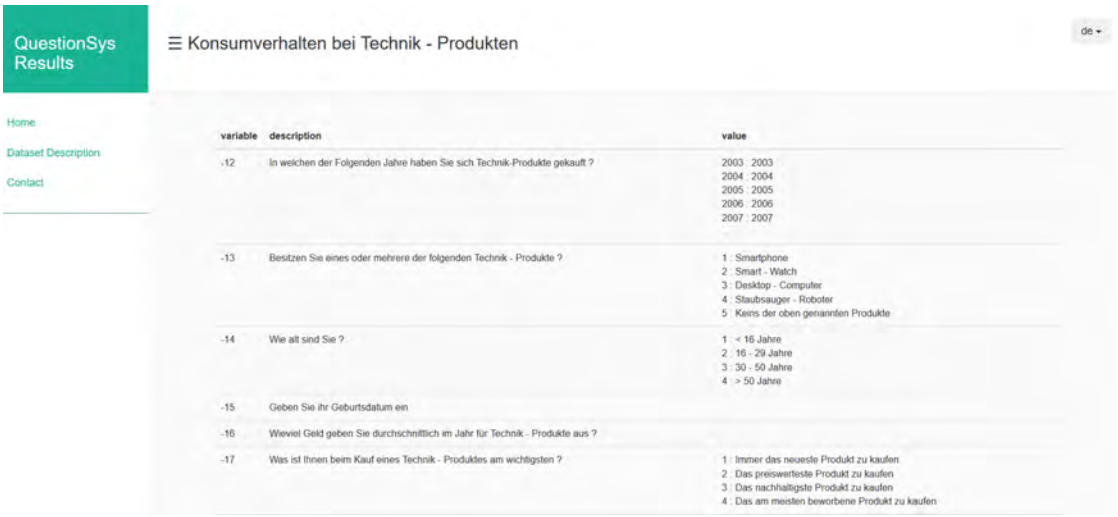


Abbildung 5.3: Anzeige der Datensatz-Beschreibung

5.1.2 Extraktion und Dekodierung

Die JSON-Daten des Fragebogens sind für den Benutzer nicht leicht zu entschlüsseln. Sie enthalten sehr viele Parameter, welche erstens der Erstellung der Fragebogen-Anwendung auf unterschiedlichen mobilen Endgeräten und zweitens der Steuerung des Ausfüll-Prozesses dienen. Auch die Ergebnisdaten besitzen einen hohen Informations- und Komplexitätsgrad. So enthält eine Liste von Ergebnissen nicht nur die Antworten auf die Fragen des Fragebogens, sondern u.a. auch Protokolldaten über dessen Ausfüll-Vorgang.

Auf diese Weise entsteht eine umfangreiche und komplexe Datenbasis. Die Extraktions- und Dekodierungsfunktion selektiert daraus nur die Antworten, und dekodiert sie, indem sie deren Inhalt mit den Strukturdaten des Fragebogens verknüpft. Diese Informationen übergibt die Extraktions- und Dekodierungsfunktion an die Benutzerschnittstelle.

5.1.3 Erstellen eines Extraktionsskripts

Nachdem der Benutzer ausgewählt hat, welche Variablen des Ergebnisdatensatzes er für seine Analyse nutzen möchte, generiert die Extraktionsplattform ein Extraktionsskript, welches die JSON-Daten in das für die Statistiksoftware benötigte Format überführt. Dieses Extraktionsskript wird vom Webserver als Datei an den Client geschickt. Der Benutzer hat die Möglichkeit es abzuspeichern oder direkt in seiner bevorzugten Entwicklungsumgebung zu bearbeiten und auszuführen.

Theoretisch könnte ein Analyst direkt aus den anonymisierten Ausführungsdaten seine Auswertungen erstellen. Dies setzt jedoch ein tiefes Verständnis des Aufbaus der Daten voraus und erzeugt einen wesentlich höheren Programmieraufwand. Um diese Arbeit zu erleichtern, führt auch das Extraktionsskript eine Selektion und Dekodierung der Ergebnisdaten durch und erfüllt damit die letzte Funktionale Anforderung FA7. Bei jedem Start des Extraktionsskripts können die JSON-Daten neu von der Web-Schnittstelle des QuestionSys Frameworks abgerufen werden, wodurch immer aktuelle Ergebnisdaten in die Analyse einfließen können.

Im Fall des in dieser Abschlussarbeit realisierten Extraktionsskripts für die Statistiksoftware R wird aus den extrahierten und dekodierten Daten eine R data.table generiert (Details dazu in Kapitel 5.2.5). Der Analyst könnte diese auch abspeichern und so seine Statistik auf einem fixierten Datensatz erstellen.

5.2 Systemkomponenten

Die Extraktionsplattform wurde mit Microsoft Visual Studio 2017 in der Community Edition entwickelt. Diese unterstützt den Entwickler u.a. bei Strukturierung, Debugging

und Anbindung an Quellcodeverwaltungssysteme. Der Quellcode wurde auf GitHub verwaltet und versioniert. Die eingesetzte Programmiersprache ist Python 3 in der Version 3.6 entsprechend dem *PEP 8 Style Guide for Python Code* [23]. Die Wahl fiel auf Python, da diese Sprache die Paradigmen der objektorientierten und der funktionalen Programmierung unterstützt. Funktionale Programmierung setzt u.a. auf die Mächtigkeit von Generatoren und Iteratoren, welche die Verarbeitung der Listen- und Dictionary-Datenstrukturen des Questionnaires wesentlich vereinfachen [24].

Python kennt sowohl Funktionen als auch Methoden. Beide verhalten sich ähnlich, unterscheiden sich jedoch darin, dass Funktionen nicht an Objekte gebunden sind. Daher können Funktionen allein durch ihren Namen aufgerufen werden, während Methoden eine Klasseninstanz benötigen [25].

Die Extraktion der Umfrageergebnisse wurde exemplarisch für die Statistiksoftware R realisiert. R ist eine interpretierte Sprache. Die benötigte Laufzeitumgebung steht auf der Projektseite des Comprehensive R Archive Network (CRAN) zum Download bereit. Die in dieser Abschlussarbeit verwendete R Version ist 3.5.0. Zur einfacheren Handhabung wurde RStudio, eine vom Unternehmen RStudio Inc. angebotene Entwicklungsumgebung und grafische Benutzeroberfläche in der Open Source Edition Version 1.1.447 eingesetzt.

Als Parser und Generator für JSON-Daten wird das R Paket `jsonlite` in der Version 1.5 vom 1. Juni 2017 eingesetzt.

Als Webframework wurde Flask in der Version 1.0.2 verwendet. Kapitel 5.2.1 beschreibt, wozu ein Webframework eingesetzt wird und wie mittels Flask die konzipierte Systemarchitektur umgesetzt wurde.

Für das Layout und die dynamischen Funktionen der HTML-Seiten wurde Bootstrap in der Version 3.3.7 eingesetzt. Bootstrap ist eine Kombination aus HTML, CSS und JavaScript.

Die nachfolgenden Kapitel beschreiben im Detail, wie die Funktionen der Extraktionsplattform in den Komponenten des Systems implementiert wurden.

5.2.1 Flask und MVC

Das eingesetzte Webframework Flask stellt als Gerüstarchitektur Funktionen für dynamische Webseiten in Form von wiederverwendbaren Klassen zur Verfügung und ist somit in der Lage, wie in Abbildung 5.4 dargestellt, die Systemarchitektur im MVC-Muster umzusetzen. Es folgt eine Einführung in Flask und MVC. Die darin vorgestellten Komponenten werden im Detail in den sich anschließenden Kapiteln beschrieben.

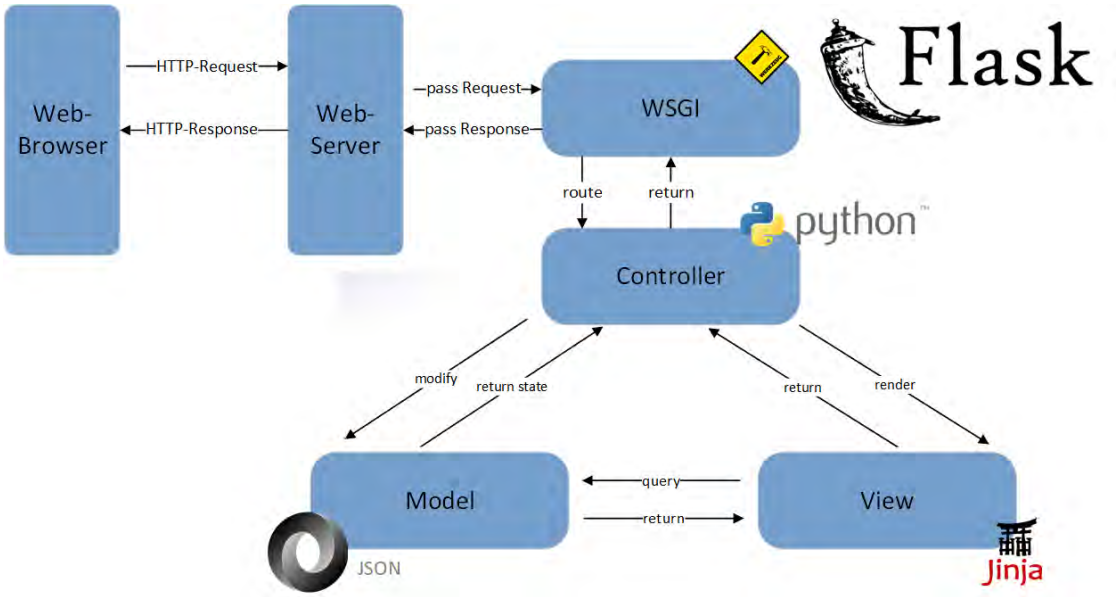


Abbildung 5.4: MVC und Flask

Das Model stellt die der Webanwendung zugrunde liegende Datenbasis dar und enthält keinerlei Informationen über deren Präsentation. Im Fall der Extraktionsplattform bildet das Model die Eigenschaften und Methoden des Fragebogens und seiner Ergebnisse ab.

Die View dient zur Präsentation des Models und enthält keine Informationen über dessen interne Struktur. Model und View kommunizieren über definierte Schnittstellen. Durch diese Eigenständigkeit und Unabhängigkeit des Models von seiner Darstellung, können seine Daten durch verschiedene Views, im Falle der Extraktionsplattform durch HTML-Seiten und Extraktionsskripte veranschaulicht werden [26]. Die Generierung dieser Views wird in Flask mit Hilfe der `Jinja2-Template Engine` realisiert [27].

Flask nutzt Python zur Implementierung des Controllers, welcher die Programmlogik beinhaltet und die Verbindung zwischen Webanwendung und dem Webserver repräsentiert.

Die Kommunikation zwischen Webserver und Python Webanwendung erfolgt über das Python Web Server Gateway Interface (WSGI). Dieses wurde 2003 von Phillip J. Eby in `PEP 333` als standardisiertes Protokoll definiert, um die Portabilität von Python Webanwendungen zwischen verschiedenartigen Webservern zu ermöglichen. Flask setzt hierzu die Utility Bibliothek `Werkzeug` ein, welche als standardisiertes WSGI die Request- und Response-Objekte für die Interaktion zwischen Webserver und Webanwendung bereitstellt [28]. Die Entwicklungsumgebung in Visual Studio nutzt ein Entwicklungs-WSGI und einen Webserver auf dem Entwicklungsrechner, der im Browser mit dem Domainnamen `localhost` angesprochen wird.

5.2.2 Controller

Der Controller der Extraktionsplattform wurde in der Python-Datei app.py realisiert. Listing 5.1 zeigt, wie ein Objekt der Klasse Flask mit dem Namen `app` instantiiert wird. Der Webserver nutzt das oben beschriebene WSGI um alle Requests des Clients an dieses Flask-Objekt zu übergeben und empfängt umgekehrt von diesem die Responses.

Das Attribut `__name__` welches an den Flask-Konstruktor übergeben wird, enthält in diesem Fall den Wert `'__main__'`, welcher in Python anzeigt, dass die Anwendung über diese Python-Datei gestartet wurde. Anhand dieses Attributs lokalisiert Flask die aufrufende Python-Datei und auch die anderen, zur Anwendung gehörenden Dateien wie `templates` und `models`.

Nach der Instantiierung des Flask-Objekts, generiert der Controller die Objekte des Models bestehend aus `questionnaire` und `results`. Dazu mehr im nächsten Kapitel.

```
1 from flask import Flask
2
3 # initialize instance of WSGI application
4 app = Flask(__name__)
```

Listing 5.1: app.py: Erzeugung des WSGI-Objekts

Gibt der Benutzer eine URL (Uniform Resource Locator) in das Adressfeld seines Browsers ein, wird diese als HTTP-Request an den Webserver geschickt, welcher sie an das WSGI weitergibt. Dieses routet die Anfrage an die entsprechende Funktion des Controllers weiter, der die verschiedenen URLs durch sog. `routes` auf unterschiedliche Webseiten oder Funktionen abbildet. Mit Hilfe der `Python Route Decorator` werden die Funktionen für die Erstellung der Views definiert, welche auch als `view functions` bezeichnet werden [29, 30].

Listing 5.2 zeigt die für die Start-Seite zuständige View-Funktion `index()`. Diese lädt die Daten des Fragebogens in der gewünschten Sprache aus dem JSON-Objekt, indem sie die entsprechende Methode des zuvor instantiierten Fragebogen-Objekts aufruft. Mit Hilfe dieser Informationen lässt sie die Ergebnisse (`results`) dekodieren und übergibt sie anschließend mittels `render_template(...)` an die Jinja2-Template-Engine zur Generierung der View. Deren Arbeitsweise wird im Kapitel 5.2.4 näher erläutert. Der Rückgabewert einer View-Funktion wird über das WSGI und den Webserver als HTTP-Response zum Webbrowser des Clients geschickt.

Die Generierung der HTML-Seite zur Anzeige der Dekodierungsinformationen durch die View-Funktion `dataset_description()` erfolgt analog.

```
1  @app.route('/')
2  def index():
3    # Load questionnaire_model data
4    # depending on required language
5    questionnaire.load_from_json(QUESTIONNAIRE_MODEL_URL_SHORT)
6
7    # Decode results using questionnaire information
8    decoded_results = m_results.DecodedResults(question_types, results, questionnaire)
9
10   return render_template('show_entries.html',
11                          decoded_results...
```

Listing 5.2: app.py: View-Funktion

Listing 5.3 zeigt die Funktion zur Generierung des R-Skripts. Ein Formular auf der HTML-Seite `show_entries.html` erzeugt einen HTTP-Request mit der URL `http://.../create-r` und gibt diesem mittels Post-Methode die Variablen mit, die der Analyst in seinem R-Skript extrahieren möchte. Die in der oben beschriebenen Utility-Bibliothek Werkzeug definierte Klasse `Headers` wird instantiiert und dient dazu, dem

HTTP-Response mitzuteilen, dass der zurückgelieferte Content ein Anhang mit dem Namen `script.r` ist. Die Antwort auf den HTTP-Request liefert schließlich ein Response-Ojekt, welches das gerenderte Template der Funktion `generate_rscript()` mittels HTTP-Response an den Client zurückschickt.

```
1  @app.route('/create_r', methods=['POST'])
2  def create_r():
3    # Check which result_values are to be transfered to R
4    if request.method == "POST":
5      selected_variables = request.form.getlist("checks")
6
7    def generate_rscript():
8    # generate RScript by rendering Jinja2 Template
9      output_from_parsed_template = render_template('output/script.r',
10               QUESTIONNAIRE_MODEL_URL=QUESTIONNAIRE_MODEL_URL,
11               RESULT_URL=RESULT_URL,
12               selected_variables=selected_variables,
13               question_types=questionnaire.question_types)
14     return output_from_parsed_template
15
16   # Add a filename
17   headers = Headers()
18   headers.set('Content-Disposition', 'attachment', filename='script.r')
19
20   # Stream the response as the data is generated
21   return Response(stream_with_context(generate_rscript()),
22           mimetype='text/r',
23           headers=headers)
```

Listing 5.3: app.py: Create R-Skript

5.2.3 Model

Das Model besteht aus den Klassen `Questionnaire`, `Results`, `DecodedResults` und `QuestionTypes`. Abbildung B.1 zeigt ein UML-Klassendiagramm mit den Klassen des Models sowie deren Attribute und Methoden. Alle Klassen außer der Helper-Klasse `DecodedResults` enthalten eine Methode `read_json()`. Diese wird beim Instantiieren der Model-Klassen aufgerufen und liest mit Hilfe der `load()` Funktion des Python Packages `json` die entsprechenden JSON-Dokumente.

Das `Questionnaire` Objekt enthält die Metadaten und Strukturinformationen des Fragebogens. Die `decoding_dictionaries` sind ein Dictionary, welches für Fragen mit vorgegebenen Antwortmöglichkeiten wiederum ein Dictionary mit den Dekodierungs-Informationen der Form `{key:displayKey}` enthält. Der `displayKey` verweist auf

die u.U. in mehreren Sprachen verfügbaren Antwortmöglichkeiten. Im Beispiel aus Kapitel 2.2 enthält dieses Dictionary für die MultipleChoice Frage mit dem key `-13` folgende Informationen:

{'12':'1' , '13':'2' , '14':'3' , '150547959606220':'4' , '1505479804089178':'5'}

Das `QuestionTypes` Objekt dient zur Steuerung der Extraktions- und Dekodierungsfunktionen.

Die Dekodierung der Fragen und ihrer Ergebnisse übernimmt eine Instanz der Klasse `DecodedResults`. Hier werden die in Kapitel 2.2.2 beschriebenen Schlüssel aus Fragebogen und Ergebnissen zusammengeführt und ausgewertet. Listing 5.4 zeigt die Methode `decode_boolean_dictionary()`, welche die Ergebnisse von Fragen mit vorgegebenen Auswahlmöglichkeiten dekodiert. Dazu benutzt die Methode den `question_key` der zu dekodierenden Frage, das oben beschriebene `decoding_dictionary` und ein Dictionary namens `result_data_value` aus dem `value`-Objekt des Ergebnisses. Im Beispiel enthält dieses Dictionary für das erste Ergebnis folgende Informationen:

{'12':True , '13':True , '14':True , '150547959606220':False , '1505479804089178':False}

Nachdem die Informationen beider Dictionaries verknüpft wurden, gibt die Methode den String `decoded`zurück. Dieser enthält die `displayKeys` der beim Ausfüllen der MultipleChoice Frage ausgewählten Antworten:

1, 2, 3

Diese Art der Dekodierung erfolgt für alle im Fragebogen enthaltenen Fragen und alle Ergebnisse. Am Ende liegen die dekodierten Informationen als Python-Liste in der `result_table` des Objekts `decoded_results` vor und kann vom Controller an die View übergeben werden.

```
1 def decode_boolean_dictionary(self, result_data_value, question_key, decoding_dictionairy):
2   """Decode every question of Type booleanDecoded (true/false vector) for displaying in table details"""
3   decoded = ''
4   for key, value in result_data_value.items():
5     if value is True:
6       #decoded concatenates all given answers to one question
7       decoded = f"{decoded}{decoding_dictionairy[key]}, "
```

```
8
9   return decoded
```

Listing 5.4: Dekodierung von Fragen mit vorgegebenen Auswahlmöglichkeiten

Die Gespräche mit Statistikern im Vorfeld des Entwurfs der Extraktionsplattform haben ergeben, dass es gängige Praxis ist, Daten in dieser kodierten Form und mithilfe einer Dekodierungstabelle zu analysieren. Daher wird die Information aus dem Beispiel nicht weiter dekodiert.

Um den Benutzer jedoch bei der Auswahl der für ihn interessanten Variablen zu unterstützen, zeigt ihm die HTML-Seite den Fragen-Text und die Dekodierungsinformationen im Klartext als `Tooltip` an, sobald er die Maus über die Frage bewegt. Die Informationen für das Tooltip stellt das Objekt `Questionnaire` als `question_array` bereit.

Tooltip

5.2.4 View

Die View dient zur Repräsentation des Models und zur Interaktion mit dem Benutzer. Dazu nutzt Flask die Funktionalität der Jinja2 Template Engine. Ein Template ist eine Datei mit einem festen Text und Platzhaltern für variablen Inhalt. Die Funktionalität, die diese Platzhalter durch die Daten des Models ersetzt, wird als `Rendering` bezeichnet. Flask stellt dazu die Methode `render_template()` bereit, welche zu Beginn der Anwendung importiert werden muss. Der Controller übergibt ihr in den View-Funktionen den Namen des Templates und eine Liste von Schlüssel-Wert Paaren, welche die aktuellen Werte der Platzhalter enthält.

Die Aufgabe der View-Funktionen ist die Erzeugung eines Response-Objekts, welches durch das WSGI zurück an den Client gesendet werden kann. Auf diese Weise können nicht nur HTML-Seiten sondern auch die Skripte zur Extraktion der Daten generiert werden.

Zur Anzeige der Result-Daten ruft der Controller die Methode `render_template()` auf und übergibt ihr u.a. den Template-Namen `show_result.html` und die `result_table` aus der zuvor beschriebenen Model-Klasse `DecodedResults`, die für jeden Result-Datensatz ein Dictionary mit `{question_key:decoded_string}` enthält.

Neben den Platzhaltern stellt Jinja2 auch Kontrollstrukturen bereit, die in den Templates die Abläufe bei der Ersetzung der Platzhalter steuern. Listing 5.5 zeigt, wie im Template `show_result.html` die HTML-Tabelle mit den Ergebnisdaten generiert wird. In zwei ineinander geschachtelten `for` Schleifen iteriert die Rendering-Methode über die Elemente der `result_table` und erstellt aus deren Werten die Datenzellen (`td`) der HTML-Tabelle.

```
1  <tbody>
2    {% for item in result_table %}
3      <tr class="results">
4        {% for value in item.values()%}
5          <td>{{value}}</td>
6        {% endfor %}
7      </tr>
8    {% endfor %}
9  </tbody>
10
```

Listing 5.5: Rendering der Ergebnis-Tabelle

5.2.5 R-Skript

Wie in Kapitel 5.2.4 beschrieben, erzeugt die Flask-Anwendung mittels Rendering ein R-Skript und schickt dieses als Anhang in einem HTTP-Response zurück an den Client. Der Benutzer kann diesen Anhang speichern oder direkt in seiner R-Entwicklungsumgebung öffnen. Das R-Skript nutzt das R-Paket `httr`, um die JSON-Daten direkt von der Webschnittstelle des Questionsys Frameworks mittels HTTP-Request anzufordern.

Ein R-Paket it eine Einheit von Code, Daten, Dokumentation und Test, die eine Aufgabe in R lösen. So profitieren R-Benutzer von erprobten und dokumentierten Lösungen für Probleme in einem bestimmten Kontext, ähnlich wie Systemarchitekten von Entwurfsmustern. Die in dieser Abschlussarbeit verwendeten R-Pakete können von der Webseite des *Comprehensive R Archive Network* (CRAN) heruntergeladen werden. Listing 5.6 zeigt, wie ein R-Paket installiert und benutzt wird. Mehrmaliges Installie-

ren und Laden eines R-Pakets innerhalb einer Session führt zu einem Fehler. Daher wird zuvor mit `require(package)` geprüft, ob es bereits verfügbar ist. Der Befehl `install.packages(package)` lädt das R-Paket von der CRAN-Webseite und installiert es in der R-Umgebung des ausführenden Rechners. Nachdem ein R-Paket installiert wurde, kann es mit `library(package)` geladen und verknüpft (`attached`) werden [31].

```
1  if(!require(package)){
2    install.packages("package")
3  }
4  library(package)
5
```

Listing 5.6: R - Installieren und Laden eines Pakets

Für das Konvertieren der JSON-Daten in ein R `data.frame` wird die Funktion `fromJSON` des R-Pakets `jsonlite` verwendet. Zur Extraktion der Fragebogen- und Ergebnisdaten durchläuft das R-Skript, ähnlich wie das Model, anhand der Fragebogen-Struktur einige Funktionen, um die Informationen der Ergebnisse zu dekodieren (siehe Dekodierungsfunktion 5.2.5). Dabei muss auch das R-Skript zeilenweise durch die komplette Liste der Ergebnisse iterieren, da es keine Möglichkeit gibt, die geschachtelten JSON-Objekte spaltenweise in eine R-Variable zu übertragen.

Dekodierungsmethode

Listing 5.7 zeigt die Methode `decode_values()`, welche die Ergebnisse jeder Frage, abhängig von ihrem Fragetyp, in eine Spalte (Variable) der weiter unten beschriebenen `data.table` dekodiert. Exemplarisch wird die Dekodierung von Fragen mit vorgegebenen Auswahlmöglichkeiten beschrieben:

Zeile 2) Der Schlüssel und der Typ der zu dekodierenden Frage sind die Eingabeparameter der Dekodierungs-Funktion.

Zeile 6) Für jede Frage wird über alle zugehörigen Ergebnisse (Datenreihen) iteriert.

Zeile 7) Wenn die Datenreihe ein zur Frage gehörendes Ergebnis enthält (! is.null()), holt sich die Funktion dessen Wert aus der Variablen `value`.

Zeile 10) Anhand des Fragetyps wird entschieden, wie dieser Wert dekodiert werden soll.

Zeile 12) Bei Fragen mit `n` vorgegebenen Auswahlmöglichkeiten ist dieser Wert ein `data.frame` mit einer Zeile und `n` Spalten. Die Spaltenüberschriften entsprechen den `keys` der Auswahlmöglichkeiten. Die zugehörigen booleschen Werte zeigen an, welche Antwort beim Ausfüllen des Fragebogens ausgewählt wurde. Durch Transposition des `data.frames` wird ein Vektor mit booleschen Werten erstellt.

Zeile 13) Dieser Vektor wird mit einer indizierten Liste der zu dieser Frage vorgegebenen Auswahlmöglichkeiten verknüpft. Das Ergebnis ist ein Eintrag in der `result_data_table`, der die Schlüssel der ausgewählten Antworten in Form eines Vektors enthält (siehe auch Abbildung 5.5a).

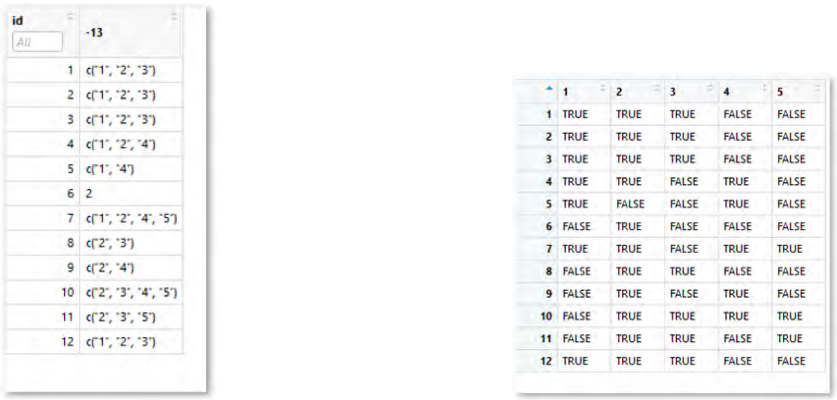
Jeder Funktionsaufruf, also jede Frage, erweitert so die `result_data_table` um eine neue Spalte.

```
1  # decode_values :
2  decode_values <- function(keyId, questionType){
3
4    j <- match(keyId,question_ss$key)
5
6    for(i in 1:row_count) {
7      if (!is.null(result_data$data$attributes$payload$results[[keyId]][[i]]$value)){
8        result_data_value <- result_data$data$attributes$payload$results[[keyId]][[i]]$value
9
10       switch(questionType,
11         boolean={
12           result_boolean <- as.vector(t(result_data_value)[,1])
13           set(result_data_table, i, eval(keyId),list(list(to_index_list(j)[result_boolean])))
14         },
15         {...},
16         set(result_data_table, i, eval(keyId), result_value)
17       )    #end switch
18     }      #end if
19   }        #end for
20 }          #end function
21
22 #result_data_table : data.table containing preselected items (keys & values)
23 result_data_table  <- data.table(id=c(1:row_count))
24
25 #k-13
26 keyId <- as.character(-13)
27 decode_values(keyId, 'boolean')
```

Listing 5.7: R - Generierung der result_data_table

Abbildung 5.5a zeigt das Ergebnis der Transformation und Dekodierung der ersten 12 Ergebnisse der aus dem Beispiel bekannten MultipleChoice Frage mit dem key `-13`. Die entsprechende Variable in der `result_data_table` enthält für jedes Ergebnis einen

Vektor mit den ausgewählten Antworten. Diese Darstellung wurde so im Gespräch mit möglichen Endanwendern vereinbart. Allerdings widerspricht sie der in den Anforderungen (Kapitel 3.4) beschriebenen sauberen R Datenstruktur. Demnach soll in einer Spalte stets nur ein Wert stehen. Um diese Anforderung umzusetzen, wird für jede Variable, die mehrere Antwortmöglichkeiten, also einen Vektor mit ausgewählten Antworten, enthält, zusätzlich eine Matrix generiert (siehe Abbildung 5.5b). Ihre Spaltenüberschriften ergeben sich aus den Schlüsseln der Antwortmöglichkeiten. Sie zeigt über einen booleschen Wert an, ob diese ausgewählt wurden.



id	-13
1	c("1", "2", "3")
2	c("1", "2", "3")
3	c("1", "2", "3")
4	c("1", "2", "4")
5	c("1", "4")
6	2
7	c("1", "2", "4", "5")
8	c("2", "3")
9	c("2", "4")
10	c("2", "3", "4", "5")
11	c("2", "3", "5")
12	c("1", "2", "3")

(a) Vektor

	1	2	3	4	5
1	TRUE	TRUE	TRUE	FALSE	FALSE
2	TRUE	TRUE	TRUE	FALSE	FALSE
3	TRUE	TRUE	TRUE	FALSE	FALSE
4	TRUE	TRUE	FALSE	TRUE	FALSE
5	TRUE	FALSE	FALSE	TRUE	FALSE
6	FALSE	TRUE	FALSE	FALSE	FALSE
7	TRUE	TRUE	FALSE	TRUE	TRUE
8	FALSE	TRUE	TRUE	FALSE	FALSE
9	FALSE	TRUE	FALSE	TRUE	FALSE
10	FALSE	TRUE	TRUE	TRUE	TRUE
11	FALSE	TRUE	TRUE	FALSE	TRUE
12	TRUE	TRUE	TRUE	FALSE	FALSE

(b) Matrix

Abbildung 5.5: R: dekodierte MultipleChoice Frage `-13`

Das R-Paket data.table

Das R-Skript stellt die dekodierten Informationen in einer `data.table` zur Verfügung. Dabei handelt es sich um eine zweidimensionale Tabelle, deren Reihen als Observationen und deren Spalten als Variablen bezeichnet werden. Sie stellt eine performantere Datenstruktur dar als das standardmäßig verwendete `data.frame` und muss ebenfalls als zusätzliches R-Paket eingebunden werden. Ein Benchmark-Test mit 10.000 Observationen und 13 Variablen zeigt, dass bei 20 Durchläufen die `data.table` im Durchschnitt um das sechsfache schneller ist, als das `data.frame` und bestätigt damit die Aussagen, die hierzu im Internet zu finden sind. Das in Listing A.1 aufgelistete R-Skript erstellt für den Benchmark-Test zunächst beide Datenstrukturen mit nur einer Variablen sowie

10.000 Observationen und fügt dann zwölf weitere Variablen hinzu. Der Grund für die schlechtere Performance des `data.frame` ist, dass bei jeder Erweiterung eine Kopie des `data.frame` erstellt werden muss, während die `data.table` für Änderungen ohne Kopie auskommt [32].

Listing 5.8 illustriert das Ergebnis des Benchmark-Tests. Dabei stellt `df` die Funktion dar, die mit dem `data.frame` arbeitet. Für die Performance der `data.table` wurden zwei Varianten (`dt` und `dt1`) getestet. Erstere erstellt die `data.table` spaltenweise, indem sie nacheinander dreizehn Spalten hinzufügt. Die zweite Variante arbeitet mit einer `data.table` die bereits beim ersten Aufruf die komplette zweidimensionale Tabelle mit allen dreizehn Spalten erstellt und dann spaltenweise füllt. Wider Erwarten ist die erste Variante im Test minimal schneller als die zweite. Eine Erklärung könnte sein, dass die zweite Variante die `data.table` mehrmals erweitern muss, da die Initialwerte der Spalten zu wenig Speicherplatz reservieren.

```
1  microbenchmark(df <- df.set(df), dt <- dt.set(dt), dt1 <- dt1.set(dt1), times=20)
2  Unit: milliseconds
3  expr         min         lq       mean     median         uq        max neval
4  df  <-   df.set(df)  2757.4957 2785.6501 2931.5664 2891.3175 2957.9916 3580.4763    20
5  dt  <-   dt.set(dt)   356.7149  365.9508  401.8023  377.3161  416.0938  602.5049    20
6  dt1 <- dt1.set(dt1)   386.7241  390.8895  403.8229  393.4901  403.9306  454.2906    20
7
8
```

Listing 5.8: Benchmark Ergebnisse

5.2.6 Python's virtuelle Umgebung

Python's virtuelle Umgebung ist ein wichtiger Bestandteil der Implementierung und der anschließenden Bereitstellung der Extraktionsplattform.

Für das Ausführen von Code benötigt Python eine Umgebung, welche aus einem Interpreter, einer Python-Bibliothek und mehreren installierten Paketen besteht. Es ist gängige Praxis, Python-Anwendungen in einer projektspezifischen virtuellen Umgebung zu entwickeln. Diese kapselt die Python-Anwendung und ihre Pakete und vermeidet Konflikte, die in einer globalen Python-Umgebung entstehen könnten. Zudem ist eine virtuelle Umgebung schlanker als eine globale, da sie nur die für die jeweilige Anwendung benötigten Python-Pakete enthält.

Die Datei `requirements.txt` gibt an, welche Python-Pakete in der virtuellen Umgebung installiert wurden. Für die Extraktionsplattform ist dies lediglich das Paket `Flask V.1.0.2`. Die komplette virtuelle Umgebung kann so aus der Quellcodeverwaltung ausgenommen werden, da sie jederzeit über `requirements.txt` neu generiert werden kann. Hierzu wird das Python Paketverwaltungsprogramm `pip` aufgerufen mit: „`pip install -r requirements.txt`“. Auf diese Weise erfolgt auch die Bereitstellung der virtuellen Umgebung auf anderen Systemen, wie z.B. einem Produktionsserver.

5.3 Erfüllung der nichtfunktionalen Anforderungen

Um die gute Lesbarkeit des Quellcodes der Extraktionsplattform zu gewährleisten, wurde diese nach dem von Guido van Rossum[3] selbst definierten *PEP 8 - Style Guide for Python Code* [23] entwickelt und trägt damit zur nichfunktionalen Anforderung **QA2** nach guter Wartbarkeit bei. Um die Übereinstimmung mit dem Style Guide sicherzustellen, wurde `Pylint` zur statischen Code-Analyse eingesetzt. `Pylint` ist ein Tool, das u.a. prüft, ob die in PEP 8 definierten Coding Standards eingehalten werden. Zudem warnt `Pylint` den Programmierer, bei möglichen fehlerhaften Code-Konstrukten, wie bspw. bei nicht initialisierten Variablen.

Die Systemarchitektur erfüllt die nichtfunktionale Anforderung **QA1** nach guter Wartbarkeit, indem sie wie in Kapitel 4.1 beschrieben, die logisch zusammenhängenden Funktionen und Daten innerhalb der Komponenten Model, View und Controller kapselt. So werden eine hohe Kohäsion und möglichst lose Kopplung realisiert.

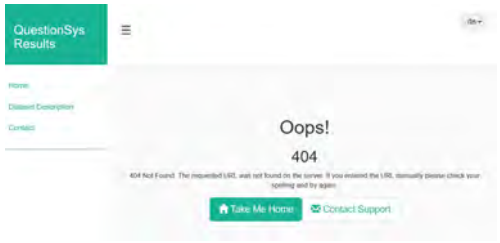
Des Weiteren erfüllt die Entkopplung der Komponenten die nichtfunktionale Anforderung **QA3** nach einfacher Erweiterbarkeit. So kann bspw. durch ein weiteres Template ein Skript für ein beliebiges Analysesystem hinzugefügt werden, welches die JSON-Daten in die dafür benötigte Datenstruktur transferiert, ohne Änderungen an den anderen Softwarekomponenten vornehmen zu müssen. Das neue Template muss hierzu lediglich im Controller hinzugefügt werden.

[3] Autor der Programmiersprache Python

Listing 5.9 zeigt, wie mit Hilfe der Flask `errorhandler decorator` die nichtfunktionale Anforderung **QA4** nach programmgesteuerter Fehlerbehandlung erfüllt wird. Die im Controller definierte Funktion `internal_server_error_404(e)` reagiert bspw. auf den Fehler `404 - page not found`. Sie veranlasst das Rendering des Templates `error.html` und schickt es wie in Abbildung 5.6a zu sehen, an den Client zurück. Auf diese Weise, werden auch interne Fehler abgefangen und der Benutzer hat die Möglichkeit, diese an einen hinterlegten Kontakt zu melden und auf die Startseite der Extraktionsplattform zurückzukehren.

```
1  @app.errorhandler(404)
2  def internal_server_error_404(e):
3    return render_template('exceptions/error.html',
4    error_code='404',
5    error_e=e), 404
6
```

Listing (5.9) Flask Errorhandler



(a) error.html

Abbildung 5.6: Fehlerbehandlung

Die graphische Benutzeroberfläche der Extraktionsplattform wurde so entwickelt, dass sie möglichst selbsterklärend ist und dadurch die nichtfunktionale Anforderung **QA5** erfüllt. Ein nicht repäsentativer Test mit acht Benutzern zeigte, dass diese innerhalb kurzer Zeit ein R-Skript wie gewünscht erstellen konnten.

Die in **QA6** geforderte Performance des Extraktionsskripts wurde berücksichtigt, indem die gemäß Benchmark Test performantere Datenstruktur `data.table` anstelle des üblichen `data.frame` zur Speicherung der transformierten Daten verwendet wird.

6

Verwandte Arbeiten

Sucht man im Internet nach *Survey Creator*, erhält man eine große Liste von Produkten zur Erstellung und Verwaltung von Online-Umfragen. Nur ein Teil davon bietet den Daten-Export der Umfrageergebnisse an. Von diesen Plattformen wiederum exportieren die meisten nur die Daten in das CSV- oder XLS-Format. Dieses Kapitel beschreibt zwei Anwendungen, von denen die erste neben CSV- und XLS-Dateien einen direkten Daten-Export für SPSS und die zweite einen Daten-Export für JSON anbietet.

6.1 SurveyMonkey

SurveyMonkey ist eine Online-Umfrageplattform des gleichnamigen US-amerikanischen Meinungsforschungsunternehmens [33]. Zum Vergleich der SurveyMonkey Funktionen mit den in dieser Abschlussarbeit beschriebenen Funktionen des QuestionSys Frameworks wurde der Beispielfragebogen in einer kostenlosen Version von SurveyMonkey generiert. Wie auch im QuestionSys Framework können Fragebogen-Anwendungen mit Hilfe eines graphischen Generators erstellt werden. Allerdings beinhaltet dieser, im Gegensatz zum QuestionSys Generator, keine Vorschau des Fragebogen-Layouts auf mobilen Endgeräten. Der Anwender muss bei der Fragebogen-Gestaltung selbst auf eine optimierte Anzeige auf mobilen Endgeräten achten. Auch bei SurveyMonkey besteht die Möglichkeit, mehrsprachige Fragebögen zu erstellen und die Ergebnisdaten anschließend in einem gemeinsamen Datensatz zu analysieren. Die in Abbildung 6.1 dargestellten Fragen der SurveyMonkey Plattform decken die gleichen Typen ab wie die im QuestionSys Framework und mit Hilfe von Logikfeatures kann auch hier das Verhalten der Fragebogen-Anwendung gesteuert werden. Da sich diese Abschlussarbeit

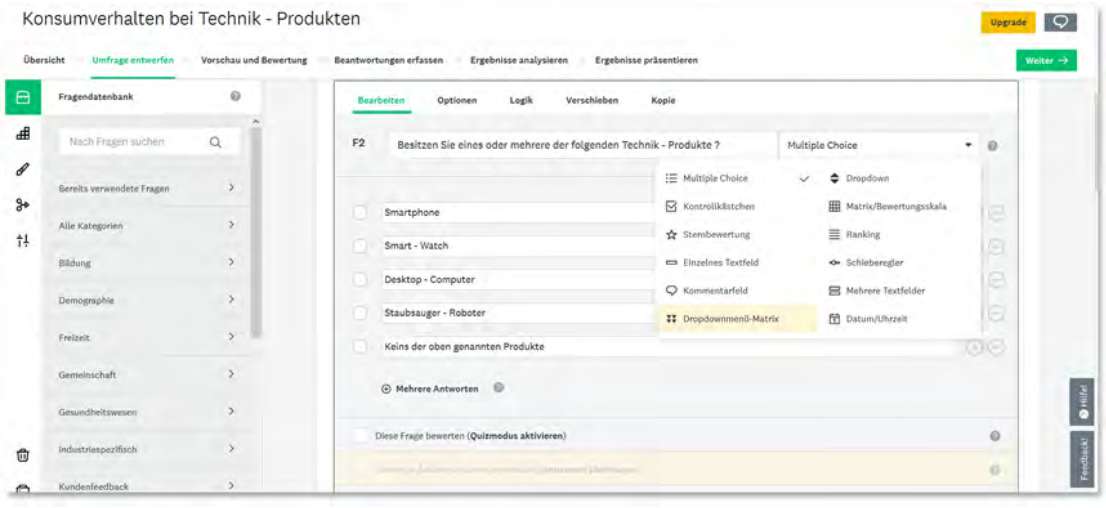


Abbildung 6.1: SurveyMonkey - Fragetypen

mit der Extraktion der Fragebogendaten beschäftigt, wird an dieser Stelle nicht weiter auf die Funktionen des Generators eingegangen.

Mit Hilfe von Filtern können in SurveyMonkey die Ergebnisdaten für die Analyse eingrenzt werden. Durch Selektion werden die Daten horizontal bspw. örtlich, zeitlich oder auf bestimmte Gruppen von Befragten eingeschränkt. Die vertikale Projektion ermöglicht wie auch im QuestionSys Framework die Eingrenzung der Ergebnisse auf bestimmte Fragen eines Fragebogens.

Zusätzlich bieten Kreuztabellen dem Benutzer die Möglichkeit, seine Ergebnisdaten in einen Kontext zu stellen, um so Abhängigkeiten in den Ausprägungen zu erkennen. Z.B. ließe sich so untersuchen, ob ein Zusammenhang zwischen Langeweile und dem Kauf von Smartphones besteht. Und schließlich lassen sich auf solche Kreuztabellen wiederum die beschriebenen Filter anwenden, um die Analyse weiter zu verfeinern [33].

Neben der Möglichkeit von Online-Analysen und Dashboards stellt SurveyMonkey die Umfrageergebnisse auch im PDF-, XLS-, CSV-, PPTX- oder SPSS-Format zum Download zur Verfügung. Ein Download für R wird nicht angeboten.

In bezahlten SurveyMonkey-Versionen können, wie auch im QuestionSys Framework, multilinguale Fragebögen erstellt werden. Zur Analyse dieser Fragebögen, werden die Ergebnisdaten aller Sprachen in einem gemeinsamen Datensatz zusammengeführt. In

diesem werden alle Fragen und Antworten, vergleichbar der Benutzerschnittstelle der Extraktionsplattform, in der Default-Sprache des Fragebogens angezeigt. Die Anzeige in einer der anderen Sprachen ist, im Gegensatz zur Extraktionsplattform, nicht möglich.

Der Download der Ergebnisdaten steht nur in bezahlten SurveyMonkey-Versionen zur Verfügung und wurde daher nicht mit dem der Extraktionsplattform verglichen.

6.2 formR

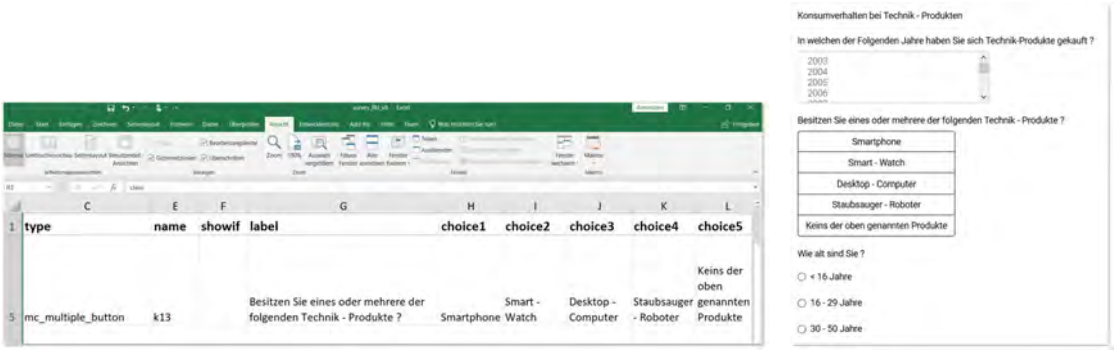
formR ist ein Open Source Framework zum Erstellen, Ausfüllen und Auswerten von Fragebögen, das von Ruben C. Arslan (Georg August Universität Göttingen) und Cyril S. Tata (GEMI, Universität Göttingen) entwickelt wurde [34]. Der Anwender kann einen Fragebogen mit den in Tabelle 6.1 aufgelisteten Fragetypen, einer Plausibilitätskontrolle und einer Steuerung des Ausfüllprozesses erstellen. Die Beantwortung des Fragebogens ist u.a. auch auf mobilen Endgeräten möglich.

Tabelle 6.1: formR Fragetypen

Familien von Fragetypen	
Simple input family	Eingaben von Text, Zahlen und email-Adressen
Sliders	Ein Wert aus einem Intervall ist auswählbar
Datetime	Auswahl oder Eingabe von Datum oder Uhrzeit
Fancy	Auswahl von GPS- oder Farb-Daten
Multiple choice	Alle Variationen von Auswahlmöglichkeiten sowohl Mehrfach- als auch Einfach-Auswahl

Die Metadaten, also Inhalt und Typ der Fragen sowie die Steuerungsinformationen werden in einem Excelsheet gespeichert. Hierbei wird der Anwender nicht wie im QuestionSys Framework durch einen graphischen Generator unterstützt. Er muss anhand von Beispielen und der formR-Dokumentation[1] selbst die korrekten Parameter für die Generierung des Fragebogens in sein Excelsheet eintragen. Beim Upload in das formR Framework wird dann die syntaktische Korrektheit der Metadaten geprüft und der Fragebogen generiert.

[1] In GitHub existiert ein ausführliches Wiki zu formR.



(a) Excelsheet (b) Survey

Abbildung 6.2: formR - Erstellung und Anzeige der MultipleChoice Frage `k13`

Abbildung 6.2 zeigt, wie die MultipleChoice Frage des Beispielfragebogens mit Hilfe eines Excelsheets in einem formR `Survey` erzeugt und dargestellt wird.

Die Fragebögen können zu sogenannten `runs` kombiniert werden, um bspw. Longitudinal-Studien oder Tagebuchstudien zu erstellen. Innerhalb eines `runs` hat der Befragte die Möglichkeit, den Ausfüllvorgang zu unterbrechen und an dieser Stelle wieder aufzunehmen. Eine email-Funktion erinnert ihn an die Teilnahme oder Fortsetzung seiner Studie.

Ein interessantes Feature des formR Frameworks ist die Einbindung von R-Skripten, -Graphiken, -Markdowns und html-widgets für R in den Fragebogen. Dadurch besteht bspw. die Möglichkeit, dem Befragten direkt beim Abschicken seiner Antworten ein visuelles Feedback zu geben. Bei der Beantwortung des Beispielfragebogens wäre es theoretisch möglich, anzuzeigen, welche Anzahlen bei den Technikprodukten der MultipleChoice Frage in Summe ausgewählt wurden. In der Praxis konnte dieses Feature nicht erfolgreich getestet werden. Laut Dokumentation ist diese Funktionalität noch in der Entwicklung [35].

Die Anzeige der Ergebnisdaten in kodierter Form ist der in dieser Abschlussarbeit beschriebenen Darstellung der Extraktionsplattform sehr ähnlich. Abbildung 6.3 zeigt, wie die Ergebnisdaten des Beispielfragebogens im formR Framework dargestellt werden. Die Variablen `created, modified` und `ended` geben Auskunft darüber, wann der

Abbildung 6.3: formR - Anzeige der Ergebnisdaten

Fragebogen ausgefüllt wurde. Den genauen Zeitstempel erhält man beim Exportieren der Daten. Die Selektions- und Filterfunktionen sind aktuell nicht aktiv.

Auch in formR kann eine Datensatzbeschreibung angezeigt und ausgedruckt werden (siehe Abbildung 6.4). Mehrsprachigkeit wird technisch nicht explizit unterstützt. Sie kann wie auch die Flexibilität des Fragebogens durch Variablen und Konditionsabfragen erreicht werden. So kann die Anzeige des Fragebogens gesteuert werden, indem ein Text nur unter bestimmten Bedingungen angezeigt wird. Um z.B. einen Fragebogen in zwei verschiedenen Sprachen zu erstellen, müssen alle Felder dupliziert werden und die Anzeige durch Bedingungsparameter gesteuert werden. Die Anzeige der Ergebnisdaten und ihrer Datensatzbeschreibung kann auf diese Weise jedoch nicht beeinflusst werden. Alle verwendeten Variablen werden als solche im Ergebnisdatensatz angezeigt und führen dazu, dass diese Anzeige sehr unübersichtlich wird.

Die Ergebnisdaten können als CSV-, Excel- oder JSON-Datei exportiert werden. Die Dokumentation empfiehlt, die Daten für R als JSON-Datei zu exportieren und anschließend mittels `jsonlite` in R zu importieren. Abbildung 6.5 zeigt das Ergebnis für den in formR erstellten Beispielfragebogen.

Zudem besteht die Möglichkeit, direkt aus R über eine API (Application Programming Interface) auf die Ergebnisdaten zuzugreifen. Das hierfür benötigte R-Paket `formr` kann von Github geladen werden.

Survey Items

type	name	label_parsed	class	showif	item_order	choices
note	note_feedback*	Konsumverhalten bei Technik - Produkten			1	
get *code*	code†				2	
select_multiple	k12†	select_multiple In welchen der Folgenden Jahre haben Sie sich Technik-Produkte gekauft ?			3	1. 2003 2. 2004 3. 2005 4. 2006 5. 2007 6. 2008 7. 2009 8. 2010 9. 2011 10. 2012 11. 2013 12. 2014
mc_multiple_button	k13*	mc Besitzen Sie eines oder mehrere der folgenden Technik - Produkte ?			4	1. Smartphone 2. Smart - Watch 3. Desktop - Computer 4. Staubsauger - Roboter 5. Keins der oben genannten Produkte

Abbildung 6.4: formR - Anzeige der Datensatzbeschreibung

	session	created	modified	ended	expired	code	k12	k13
1	NA	2018-10-16 20:38:27	2018-10-16 20:40:16	2018-10-16 20:40:16	NA		9, 12	1, 2, 3
2	NA	2018-10-16 20:45:04	2018-10-16 20:46:22	2018-10-16 20:46:23	NA		1, 2, 3, 4	1, 3, 4

Abbildung 6.5: formR - Anzeige der Datensatzbeschreibung

7

Zusammenfassung und Ausblick

In dieser Abschlussarbeit wurde eine Webanwendung entwickelt, mit deren Hilfe die Ergebnisdaten einer Fragebogen-Anwendung des QuestionSys Frameworks komfortabel angezeigt und für die Auswertung in einem Analysesystem extrahiert werden können. Die entstandene Extraktionsplattform erspart dem Analysten eine sehr zeitaufwendige Einarbeitung in die Techniken zur Transformation der Fragebogendaten und in deren komplexe Datenstrukturen.

Als Grundlage für die Konzeption der Extraktionsplattform wurden das Speicherformat und die Struktur der Ergebnisdaten im Hinblick auf die Selektion und Dekodierung der Informationen für die statistische Analyse untersucht. Die Bereitstellung der Daten wurde in einem ersten Schritt für die Statistik-Software R realisiert. Daher mussten auch die technischen Voraussetzungen für die Transformation der von der QuestionSys-Schnittstelle bereitgestellten JSON-Dokumente in das von R verwendete Datenformat analysiert werden.

Um die Anforderungen an die Extraktionsplattform spezifizieren zu können, wurde ein experimenteller Prototyp entwickelt, welcher iterativ evaluiert und verfeinert wurde. So konnten anhand eines Beispielfragebogens und durch Befragung von potentiellen Nutzern die funktionalen und nichtfunktionalen Anforderungen an die zu entwickelnde Extraktionsplattform ausgearbeitet werden.

Auf Basis der Erkenntnisse aus dem experimentellen Prototyping wurde die Softwarearchitektur der Extraktionsplattform konzipiert. Dabei wurde besonderer Wert auf die Erweiterbarkeit der Anwendung gelegt, um zusätzliche Analysesysteme oder Änderungen der Ergebnisdaten integrieren zu können, ohne das Gesamtkonzept überarbeiten zu müssen.

Die anschließende Implementierung der Extraktionsplattform erfolgte in Python und mit dem Einsatz des Python Frameworks Flask. Letzteres stellt u.a. eine Template-Engine zur Verfügung, mit deren Hilfe nicht nur dynamische Webseiten sondern auch die Skripte zur Extraktion der Ergebnisdaten generiert und an den Client übertragen werden können. Zur Sicherung der Softwarequalität wurden analytische und konzeptionelle Maßnahmen eingesetzt. Dabei wurden u.a. eine statische Codeanalyse, ein Benchmark-Test und eine Befragung zur Benutzerfreundlichkeit durchgeführt, um zur Erfüllung der nichtfunktionalen Anforderungen beizutragen.

Während der Entwicklung, der Validierung und der vergleichenden Analyse mit verwandten Arbeiten ergaben sich interessante Aspekte für mögliche Weiterentwicklungen der Extraktionsplattform. Sie werden in den nachfolgenden Abschnitten erläutert und schließen damit diese Arbeit ab.

7.1 Weiterentwicklung

Die bei der Implementierung und den anschließenden Tests gewonnenen Erkenntnisse führten zu den nachfolgenden Überlegungen bezüglich der Weiterentwicklung der Extraktionsplattform und der Schnittstelle zum QuestionSys Framework.

Anbindung weiterer Analysesysteme

Der Einsatz der im Kapitel 5.2.4 beschriebenen Jinja-Templates ermöglicht eine komfortable Erweiterung der Extraktionsplattform hinsichtlich der Einbindung weiterer Analysesysteme. Hierzu muss ein Skript oder Makro entwickelt werden, das im jeweiligen Analysesystem die Daten aus den JSON-Dokumenten der QuestionSys-Schnittstelle extrahiert und dekodiert. Diese Lösung birgt die Gefahr, dass bei einer Änderung der QuestionSys-Schnittstelle alle Extraktionsskripte angepasst werden müssten. Allerdings ist dies ein grundsätzliches Problem bei allen Schnittstellenänderungen.

Es wäre interessant, zu untersuchen, ob die Selektion und die Dekodierung der Daten serverseitig in der Web-Schnittstelle des QuestionSys Frameworks erfolgen könnten. Das hätte den großen Vorteil, dass die übertragenen JSON-Dokumente wesentlich

kleiner wären und die Logik zur Dekodierung der Daten nur an einer Stelle implementiert und geändert werden müsste. Die Transformation der JSON-Daten in das für das jeweilige Analysesystem benötigte Format könnte weiterhin durch clientseitige Skripte durchgeführt werden, um so die zentrale Schnittstelle zu entkoppeln.

Funktionserweiterung um Projektionen

Die implementierte Extraktionsplattform ermöglicht die Selektion der Variablen, um so den Analysedatensatz *vertikal* einzuschränken. Die Möglichkeit einer Projektion, welche die Menge der Analysedaten *horizontal* einschränken könnte, wäre wünschenswert. Allerdings müsste zunächst untersucht werden, ob diese Einschränkung der Analysedaten bereits durch die QuestionSys-Schnittstelle auf dem Server oder erst durch ein entsprechendes Skript auf dem Client durchgeführt werden sollte.

Weitere Fragetypen

Die Erweiterung der Extraktionsplattform um die Dekodierung neuer Fragetypen muss an mehreren Stellen im Programmcode erfolgen. Durch den Einsatz des MVC-Musters wurde der Umfang der Änderungen in der Benutzerschnittstelle der Webanwendung auf die Model-Klasse beschränkt. Allerdings gilt dies nicht für die clientseitig ausgeführten Extraktionsskripte, was ein weiteres Argument für eine zentrale Dekodierung durch die QuestionSys-Schnittstelle liefert.

Literaturverzeichnis

[1] Kuckartz, U., Rädiker, S., Ebert, T., Schehl, J.: Fehlende Werte, immer ein Problem. In: Statistik: Eine verständliche Einführung, Wiesbaden, Springer (2013) 20–21

[2] van Gelder, M.M.H.J., Bretveld, R.W., Roeleveld, N.: Web-based Questionnaires: The Future in Epidemiology? In: American Journal of Epidemiology. (2010) 1292–1298

[3] Schobel, J., Pryss, R., Schickler, M., Ruf-Leuschner, M., Elbert, T., Reichert, M.: End-User Programming of Mobile Services: Empowering Domain Experts to implement Mobile Data Collection Applications. In: 2016 IEEE International Conference on Mobile Services (MS), IEEE (2016) 1–8

[4] Bray, T.: The JavaScript Object Notation (JSON) Data Interchange Format. RFC 8259, IETF (2017) https://tools.ietf.org/rfc/rfc8259.txt.

[5] Marrs, T.: JSON at Work: Practical Data Integration for the Web. O'Reilly Media, .Inc (2017)

[6] Rexer Analytics: Data mining software. In: 5th Annual Data Miner Survey. (2011) 13–18

[7] Naumann, S., Hohagen, F.: Parsing: Eine Einführung in die maschinelle Analyse natürlicher Sprache. Leitfäden und Monographien der Informatik. Vieweg+Teubner Verlag (2013)

[8] Dasu, T., Johnson, T.: Exploratory Data Mining and Data Cleaning. Wiley Series in Probability and Statistics. Wiley (2003)

[9] Ludewig, J., Lichter, H.: Vorgehensmodelle - prototyping. In: Software Engineering: Grundlagen, Menschen, Prozesse, Techniken, Heidelberg, dpunkt (2010) 163–168

[10] Floyd, C.: A Systematic Look at Prototyping. In: Approaches to Prototyping, Berlin, Heidelberg, Springer (1984) 1–18

[11] Sommerville, I.: Software Engineering. Pearson Education (2012)

[12] ISO: International Standard ISO/IEC/IEEE 29148:2011(E): Requirements engineering. (2011)

[13] Ooms, J.: The jsonlite package: A practical and consistent mapping between json data and r objects. arXiv preprint arXiv:1403.2805 (2014)

[14] R Core Team: An Introduction to R. R Foundation for Statistical Computing. (2018)

[15] Codd, E.: The Relational Model for Database Management: Version 2. Addison-Wesley, Boston (1990)

[16] Wickham, H.: Tidy data. Journal of Statistical Software (2014)

[17] Pichler, R., Roock, S.: Agile Entwicklungspraktiken mit Scrum. dpunkt-Verlag, Heidelberg (2011)

[18] Martin, R.: Clean Architecture: Das Praxis-Handbuch für professionelles Softwaredesign.Regeln und Paradigmen für effiziente Softwarestrukturierung. MITP Verlags GmbH, Frechen (2018)

[19] Gharbi, M., Koschel, A., Rausch, A., Starke, G.: Basiswissen für Softwarearchitekten. dpunkt, Heidelberg (2017)

[20] Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Pearson Education (2009)

[21] Diepenmaat, J., Veer, R.v.: The Model View Controller pattern in web applications. https://zeekat.nl/articles/mvc-for-the-web.html (2018) [Online; Stand 29. September 2018].

[22] Statista: Statistik-Lexikon: Definition Variable. https://de.statista.com/statistik/lexikon/definition/137/variable/ (2018) [Online; Stand 7. Okt. 2018].

[23] van Rossum, G., Warsaw, B., Coghlan, N.: Style guide for python code. https://www.python.org/dev/peps/pep-0008/ (2001) [Online; Stand 24. August 2018].

[24] Cassell, L., Gauld, A.: Python Projects. Wrox (2014)

[25] Sohail, A.: Difference between method and function in python. https://www.geeksforgeeks.org/difference-method-function-python/ (2018) [Online; Stand 22. Oktober 2018].

[26] Warin, G.: Mastering Spring MVC 4. Packt Publishing (2015)

[27] Maia, I.: Building Web Applications with Flask. Packt Publishing (2015)

[28] Eby, P.J.: Python web server gateway interface v1.0. https://www.python.org/dev/peps/pep-0333/#rationale-and-goals (2003) [Online; Stand 24. August 2018].

[29] Guardia, C.d.l.: Python Web Frameworks. O'Reilly Media, Inc. (2016)

[30] Grinberg, M.: Flask Web Development, 2nd Edition. O'Reilly Media, Inc. (2018)

[31] Wickham, H.: R Packages. O'Reilly Media, Inc. (2015)

[32] Dowle, M.: data.table: Assignment by reference. https://www.rdocumentation.org/packages/data.table/ (2018) [Online; Stand 08. September 2018].

[33] SurveyMonkey Europe UC: Surveymonkey. (https://de.surveymonkey.com/) [Online; Stand 22. Oktober 2018].

[34] Arslan, R., Tata, C., Walther, M.: formr: A study framework allowing for automated feedback generation and complex longitudinal experience sampling studies using r. (https://formr.org) [Online; Stand 28. Oktober 2018].

[35] Arslan, R.: formr: How to create feedback. (https://github.com/rubenarslan/formr.org/wiki/How-to-create-feedback) [Online; Stand 28. Oktober 2018].

A

Quelltexte

Listing A.1 zeigt den Code für den Benchmark-Test zur Auswahl der R-Datenstruktur `data.table`.

```
1  ########################################
2  # Benchmark for data.table vs data.frame
3  ########################################
4  if(!require(data.table)){
5  install.packages("data.table")
6  }
7  if(!require(microbenchmark)){
8  install.packages("microbenchmark")
9  }
10
11 library(data.table)
12 library(microbenchmark)
13
14 rm(list=ls())
15 gc()
16
17 n <- 10^4
18
19 df <- data.frame(id=c(1:n))
20 dt <- data.table(id=c(1:n))
21 dt1 <- data.table(a=c(1:n), b=0,c=0,d=0,e=0,f=0,g=0,h=0,i=0,j=0,k=0,l=0)
22 cn <- colnames(dt1)
23
24 a <- NA
25 b <- NA
26 c <- NA
27 d <- NA
28 e <- NA
29 f <- NA
30 g <- NA
31 h <- NA
32 i <- NA
33 j <- NA
34 k <- NA
35 l <- NA
36
37
38 df.set <- function(df) {
39 rm(df)
40 gc()
41 df <- data.frame(id=c(1:n))
```

```
42 df <- cbind(df, a)
43 df <- cbind(df, b)
44 df <- cbind(df, c)
45 df <- cbind(df, d)
46 df <- cbind(df, e)
47 df <- cbind(df, f)
48 df <- cbind(df, g)
49 df <- cbind(df, h)
50 df <- cbind(df, i)
51 df <- cbind(df, j)
52 df <- cbind(df, k)
53 df <- cbind(df, l)
54 for (j in 2:length(df)){
55 for(i in 1:n) {
56 df[[j]][i] <- i
57 }
58 }
59 return(df)
60 }
61
62
63 dt.set <- function(dt) {
64 rm(dt)
65 gc()
66 dt <- data.table(id=c(1:n))
67 for (j in (cn)){
68 for(i in 1:n) {
69 set(dt,i,j, i )
70 }
71 }
72 return(dt)
73 }
74
75
76
77 dt1.set <- function(dt1) {
78 rm(dt1)
79 gc()
80 dt1 <- data.table(id=c(1:n), a=0,b=0,c=0,d=0,e=0,f=0,g=0,h=0,i=0,j=0,k=0,l=0)
81 for (j in (cn)){
82 for(i in 1:n) {
83 set(dt1,i,j, i )
84 }
85 }
86 return(dt1)
87 }
88
89
90
91 m <- microbenchmark(df <- df.set(df), dt <- dt.set(dt), dt1 <- dt1.set(dt1), times=10)
```

Listing A.1: data.table Benchmark

B

Abbildungen

Abbildung B.1: Model-Klassen

Abbildung B.2: Transformation der Ausführungsdaten von JSON nach R

Abbildungsverzeichnis

Tabellenverzeichnis

Name: Rita Unseld Matrikelnummer: 864300

Erklärung

Ich erkläre, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den .....

Rita Unseld